2 AMIGA Markt&Technik

DM 16,–
ÖS 120,– /Sfr 16,–
Lit 14000 /hfl 21,– /dkr 75,–

SONDERHEFT

Animation mit Nachbrenner

- *Zum Abtippen: Sensationelles Tool für bewegte Grafiken*

3D

Super-Listings

- *Fast-Load-Copy: Schnelles Directory durch optimierte Kopien*
- *Haushaltsbuch: Ihre Finanzen endlich im Griff*
- *Gravitation: Jonglieren mit Sonnen und Planeten*
- *Vokabeltrainer: Intelligentes Lernprogramm für mehrere Sprachen*

Tips & Tricks zu Superbase

Alle Programme auch auf Diskette erhältlich

Ihre Meinung bitte

Alle, die an der Entstehung des Amiga-Sonderheftes mitwirken, haben stets ein Ziel vor Augen: Wir wollen Ihnen den Umgang mit Ihrem Computer erleichtern. Sie sollen in jedem Heft viele interessante Themen finden. Die vorgestellten Programme schließen Lücken im Software-Angebot, sind eine interessante und erschwingliche Alternative zu kommerziellen Produkten. Die Schwerpunkte dieses Sonderhefts sind nach langen Diskussionen ausgewählt worden. Die Texte, die Sie auf den folgenden Seiten lesen werden, sind überprüft, geändert, nochmals überprüft und nochmals geändert worden. Die Entstehung des Heftes, das Sie jetzt in den Händen halten, hat ungefähr drei Monate gedauert. Sind Sie mit dem zweiten Amiga-Sonderheft zufrieden, haben wir die richtige Themenwahl getroffen? Sagen Sie uns bitte Ihre Meinung. Gelegenheit dazu geben wir Ihnen mit der Leserumfrage auf Seite 14. Schreiben Sie uns, welche Themen Sie besonders interessieren. Wünschen Sie sich in den folgenden Sonderheften Kurse und Listings? Sollen wir auf lange Listings im Heft völlig verzichten, sie lieber nur auf der Programmservice-Diskette veröffentlichen?



Ihre Meinung ist für uns wichtig, um in kommenden Ausgaben noch interessantere Themen für Sie auszuwählen. Übrigens: Die Sonderhefte werden ab 1989 regelmäßig im Abstand von zwei Monaten erscheinen.

Aber zusätzlich zum Fragebogen steht Ihnen natürlich jederzeit frei, per Brief, Postkarte oder telefonisch Stellung zu den Sonderheften zu nehmen und Fragen an die Redaktion zu stellen.

Wenn Sie anrufen möchten, beachten Sie bitte unsere regelmäßige Sprechzeit: An jedem Donnerstag sind wir von 14 bis 17 Uhr zu erreichen. Gemeinsam mit Ihnen haben wir die Chance, sehr hohen Ansprüchen zu genügen. Nutzen Sie diese Möglichkeit, wir geben unser Bestes.

Ihr
Ralf Sablowski
Redakteur

Ralf Sablowski

Behalten Sie Einnahmen und Ausgaben mit dem Amiga unter Kontrolle. Mit dem elektronischen Haushaltsbuch bekommen Sie Ihre Knete in den Griff — und das mit viel Komfort.

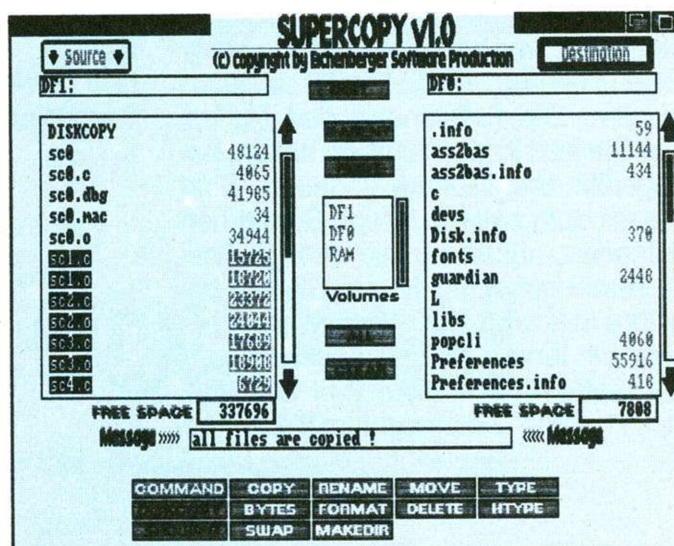
SEITE **18**





In diesem Spiel geht es um Karten, Kontinente und Kanonen. Erobern Sie die Spielwelt mit Taktik und Glück.

SEITE **10**



Der Umgang mit dem CLI hält nicht, was die grafische Benutzeroberfläche des Amiga verspricht. Führen Sie CLI-Operationen von nun an bequem mit »Supercopy« durch.

SEITE **90**

INHALT

Alle Programme aus Artikeln mit einem -Symbol finden Sie auch auf der Programmservice-Diskette zu diesem Sonderheft

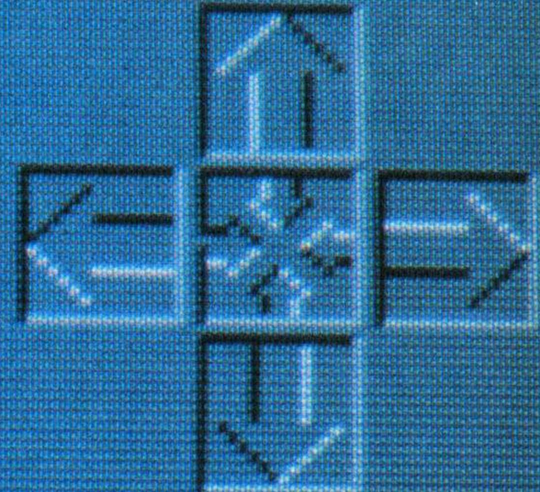
Mit dem Objekt-Editor erstellen Sie Sprites für eigene Spiele — Animation inklusive.
SEITE **6**

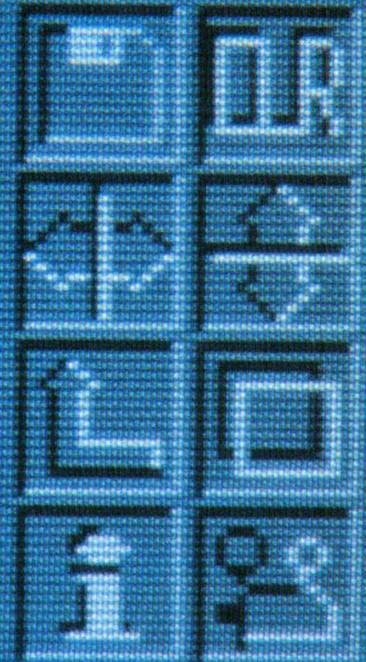
Beteiligen Sie sich an der Leser-Umfrage und gewinnen Sie ein Doppellaufwerk mit Overdrive-System-Port.
SEITE **14**

Sprite-Editor

Sprites gehören zu Spielen wie der Monitor zum Computer. Leider müssen diese auch auf dem Amiga in mühevoller Kleinarbeit erstellt werden. Unser Sprite-Editor beseitigt dieses Problem und macht selbst das Entwickeln von überdimensionalen Sprites zum Kinderspiel.







Objekt Nr

001

R G B

Es gibt kaum Spiele, die ohne Sprites auskommen. Dabei beschränkt sich die Verwendung von bewegten Objekten keineswegs nur auf Actionspiele. Auch Brettspiele, Knobelspiele, Adventures etc. profitieren ungemein durch den Einsatz beweglicher Objekte. Mit der mausgesteuerten Benutzeroberfläche des »Amiga-Object-Editor« (Bild 1) lassen sich Sprites und Animationssequenzen schnell und mühelos erstellen und editieren. Auch DPaint-Brushes lassen sich damit bearbeiten.

Geben Sie den DATA-Lader »AOE__Gen« (Listing 1) mit dem Checksummer ein. Da das Programm einen relativ hohen Speicherbedarf hat, muß vor der Eingabe des Listings im Direktmodus des Basic-Editors »CLEAR, 150000&« eingegeben werden. Dies hat auch dann zu geschehen, wenn »AOE__Gen« erneut in den Speicher geladen wird. Nach dem Start des Laders generiert dieser das lauffähige Programm »AOE__V1.2« auf Diskette. Nach diesem Vorgang, der etwas Zeit beansprucht, kann der Object-Editor gestartet werden. Dabei ist zu beachten,

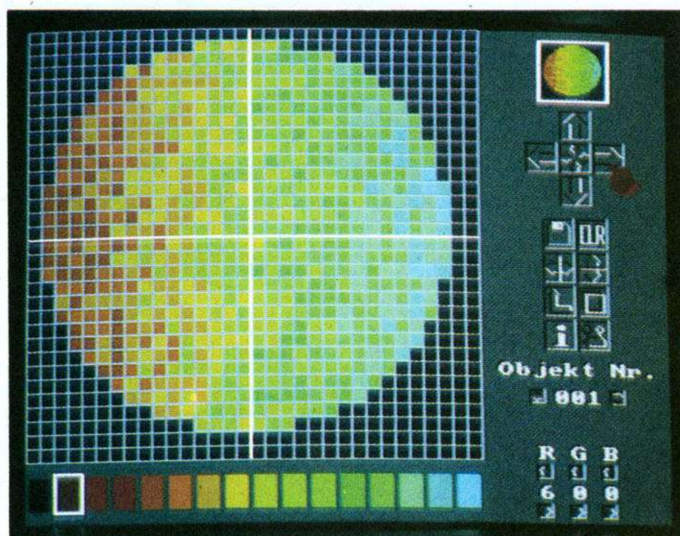


Bild 1. Die Benutzeroberfläche des »Amiga-Object-Editors« mit den verschiedenen Icons

daß das vom Lader generierte File nur vom CLI aus gestartet werden kann, obwohl ein Icon vorhanden wäre.

Sie werden nach dem Start aufgefordert, die Anzahl der Objekte, die bearbeitet werden sollen, einzugeben. Klicken Sie dazu das Eingabefeld an und tragen die gewünschte Objektanzahl ein.

Das Zeichnen eines Objektes ist durch die Maussteuerung sehr einfach. Es stehen 16 verschiedene Farben zur Verfügung, die durch einfaches Anklicken ausgewählt werden. Die Rot-Grün- und Blauanteile der gewählten Farbe können durch die Schalter am unteren rechten Bildschirmrand verändert werden.

Gezeichnet wird ein Sprite mit Hilfe der Maus, wobei mit der linken Maustaste Punkte gesetzt und mit der rechten gelöscht werden. Das zu bearbeitende Sprite wird bei »Objekt Nr.« ausgewählt.

Im Zeichenfenster wird das Riesensprite, das in Wirklichkeit aus vier Einzelsprites besteht, stark vergrößert dargestellt. Rechts oben sieht man das Sprite in Originalgröße. Unter diesem Fenster befinden sich vier Pfeile. Durch Anklicken dieser Pfeile kann das aktuelle Sprite in alle vier Himmelsrichtungen verschoben werden. Klicken Sie das Icon in der Mitte der Pfeile an, so ändern diese ihre Richtung und das Sprite wird nun bei der Verwendung der Pfeilsymbole in Richtung seiner Mittelachse gestaucht. Unter den Pfeil-Icons befinden sich acht weitere Icons, auf deren Funktionen nun eingegangen werden soll:

Icon 1 (Disketten-Icon): Hier haben Sie die Möglichkeit, Sprites zu laden und zu speichern. Zum Laden muß die entsprechende Option gewählt und die Nummer, unter der das Objekt im Speicher stehen soll, sowie der Filename eingegeben werden.

Adresse	Wert	Bemerkung
$00000000	"AOE ".L	Kennung der Objektdatei
$00000004	0000.W	Anzahl der Objekte in der Datei
$00000006	0000.W	Nun folgen 16 Wörter mit der Farbinfomation
$00000026		Es folgen die Objektdaten: Anzahl der Objekte mal 640 Byte.
Die Objektdaten haben folgenden Aufbau:		
1. Objekt:		1. Bitplane (128 Byte)
		.....
		4. Bitplane (128 Byte)
		5. Maske (128 Byte)
2. Objekt:		1. Bitplane (128 Byte)
		.....
Jedem gesetzten Bit in der Maske entspricht ein gesetzter Punkt im Sprite		

Tabelle 1. Der Aufbau eines Files im »AOE«-Format

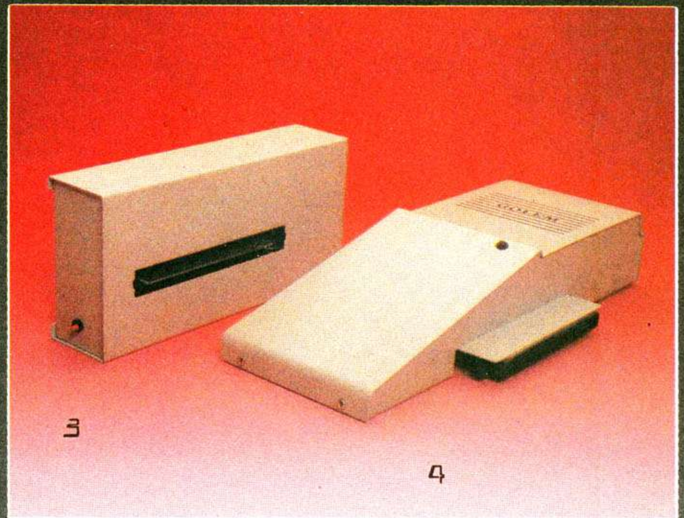
Fortsetzung des Artikels auf Seite 126

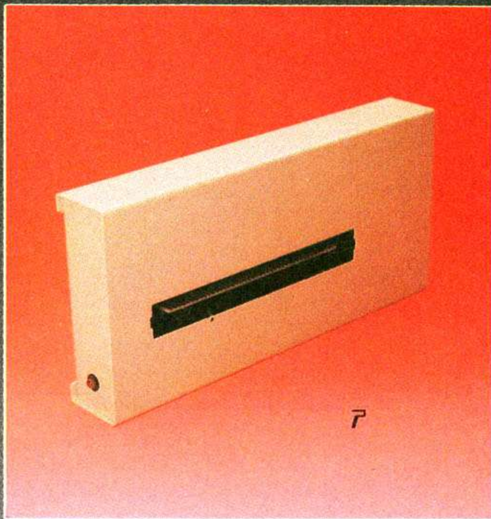
KUPKE

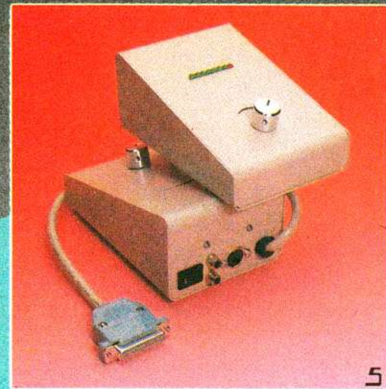


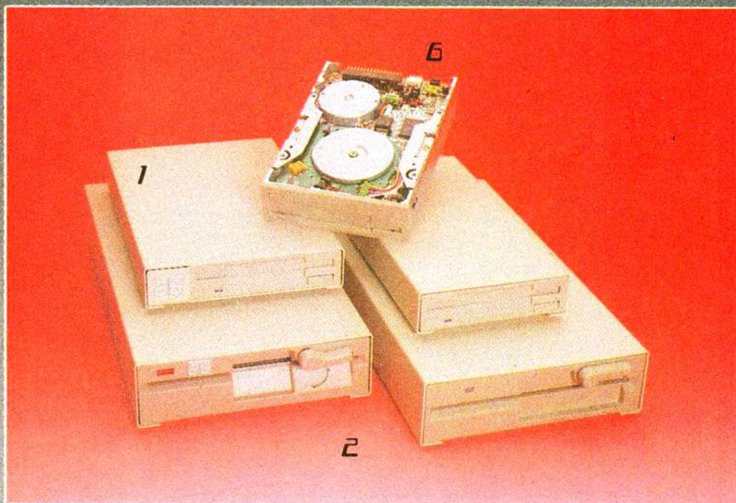
02 31/81 83 25-27
Telefax 02 31/81 74 29
D-4600 Dortmund 1
Burgweg 52 a

GOLEM









Wir liefern im 3-Tage-Rhythmus

1 Golem Drive 3,5 Display

NEC 1037a mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalter ● Busdurchführung bis DF3 ● Sidecar, PC 1 und PC Karten-kompatibel ● Trackdisplay zur aktuellen Spur- und Kopfanzeige

mit **Display** **DM 359.-**
ohne **Display** **DM 339.-**

2 Golem Drive 5,25 Display

NEC Laufwerk mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalter 40/80 Track-Umschalter ● Busdurchführung bis DF3 ● PC Karten, Sidecar und PC 1 kompatibel ● Trackdisplay zur aktuellen Spur- und Kopfanzeige

mit **Display** **DM 449.-**
ohne **Display** **DM 419.-**

3 Golem Ram Box 2MB

2MB Speichererweiterung für dem Amiga 1000 ● ansteckbar am Systembus ● Abschalter ● Busdurchführung ● autokonfigurierend ● Betriebskontrollanzeige ● Amiga-farbenes Metallgehäuse ● erweitert den Grundspeicher auf 2,5 Megabyte

komplett **DM 1398.-**
ohne Ram's **DM 449.-**

4 Golem 500 Ram Box

2MB Speichererweiterung im formschönen 500'er Design ● Busdurchführung ● autokonfigurierend ● Betriebskontrollanzeige ● externer Anschluß an den Systembus ● erweitert den Grundspeicher auf 2,5 Megabyte

komplett **DM 1398.-**
ohne Ram's **DM 449.-**

Technische Änderungen vorbehalten

5 Golem Sound Stereo

Audio Digitizer der Spitzenklasse ● kompatibel zur meisten Samplersoftware ● DIN- und Cinch Anschluß auch für Micro geeignet ● optisches Aussteuerungsdisplay ● Stereowandlung ● umschaltbar auf Mono-Betrieb

Stereo **DM 189.-**
Mono ohne Display **DM 139.-**

6 Golem Drive A 2000

internes Amiga Drive ● NEC 1036a mit heller Frontblende ● einbaufertig modifiziert ● mit Staubschutzklappe ● incl. Einbauanleitung und Montagesatz

DM 200.-

7 Kickstart/Uhrenmodul

"Bitte Workbench einlegen", meldet ihr Amiga 1000 nach dem Einschalten mit dem extern ansteckbaren Kickstartmodul ● Busdurchführung ● Abschalter, so daß andere Kickstartversionen wieder gebootet weden können ● alle gängigen Kickstart-Versionen lieferbar

DM 199.-

Amiga 500/2000 kompatibeles Uhrenmodul ● Akkugepuffert ● extern ansteckbar

im Extragehäuse **DM 149.-**
Uhr u. Kick in einem Gehäuse **DM 299.-**

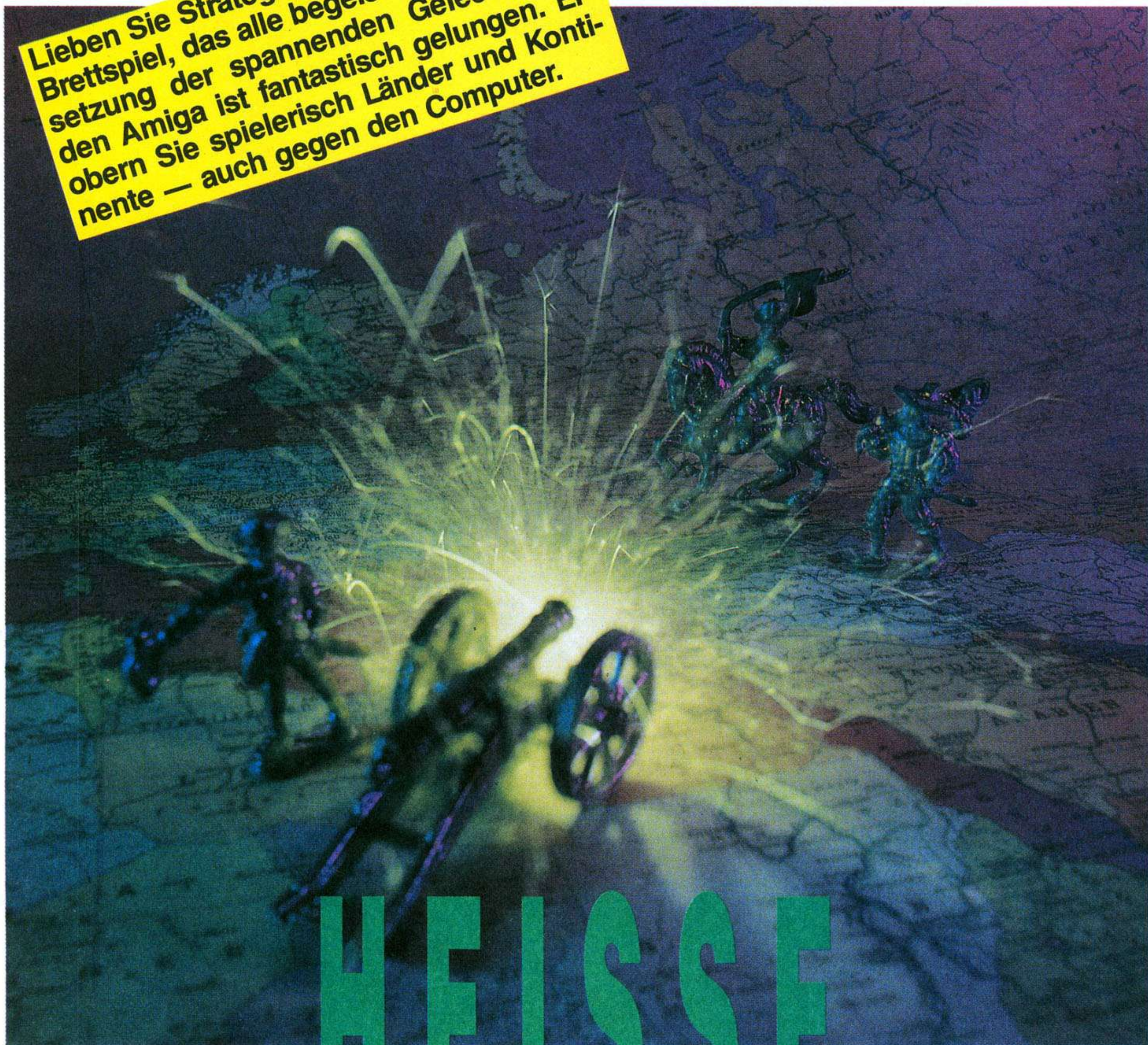
8 Kickstartumschaltplatine

Intern einsteckbare Umschaltplatine bestückt mit einem zusätzlichen Kickstart ● alle gängigen Versionen ● keine Lötarbeiten erforderlich ● umschaltbar auf original Kickstart

komplett **DM 149.-**
ohne Eprom's **DM 59.-**

Entwurf + Druck PLUM, Düren-Oberzier, TEL. 02428/711

Lieben Sie Strategiespiele? Risiko ist ein Brettspiel, das alle begeistert. Diese Umsetzung der spannenden Gefechte auf den Amiga ist fantastisch gelungen. Erobern Sie spielerisch Länder und Kontinente — auch gegen den Computer.



HEISSE Schlachten im Amiga

Die Blauen sind auf dem Vormarsch. Australien wurde bereits im Handstreich erobert, Asien ist zu zwei Dritteln von ihnen besetzt. Die Gelben haben sich in Nord-Amerika eingenistet, auch sie sind eine große Bedrohung. Der Gegner konzentriert seine Armeen im Mittleren Osten. Jetzt greift der Feldherr der blauen Steine mit einer gewaltigen Armee Europa an, Ihre einzige Bastion wackelt. Werden Sie diesen gewaltigen Zweikampf für sich entscheiden? So könnte die Lage im Verlaufe des Strategiespiels »Amiga gegen den Rest der Welt« aussehen. Diese Umsetzung eines beliebten Brettspiels der Firma Parker ist faszinierend. Jeder, der einmal

Amiga gegen die Welt

mit ein paar Freunden um die Welt gestritten hat, wird das bestätigen. Natürlich kann auch Ihr Amiga direkt in den Kampf eingreifen. Bei maximal sechs Mitspielern in einem Spiel sind bis zu fünf Computer-Gegner zu aktivieren.

Wenn Sie in den Genuß des Strategiespiels kommen wollen, geben Sie bitte Listing 1 (»Lader«) und das Hauptprogramm (Listing 2) mit dem Checksummer ein. Hinweise zu dieser Eingabehilfe finden Sie auf Seite 159 in diesem Heft.

Haben Sie die Programme auf Diskette gespeichert, kann der Spaß beginnen. Laden Sie Amiga-Basic und starten von dort aus das Programm »Lader«. Der Speicher für das Basic-Programm wird jetzt auf 40 000 Byte vergrößert und das Hauptprogramm geladen.

Die erste Meldung des Spiels fragt nach der Teilnehmerzahl der menschlichen Spieler. Wählen Sie eine Zahl von 0 bis 6, schließen Sie diese und alle folgenden Eingaben bitte mit <RETURN> ab. Jetzt möchte das Programm noch wissen, wie viele Gegner der Computer übernehmen

soll. Die erlaubten Eingaben sehen Sie in der Klammer hinter der Frage.

Anschließend sind für die Spieler (menschlich) Namen einzugeben, maximal acht Buchstaben sind jeweils erlaubt. Die Wahl der Namen für die computer-gesteuerten Mitspieler übernimmt das Basic-Programm. Sie lauten beispielsweise »Amiga« und »Atari«, der immerwährende Konkurrenzkampf zwischen diesen beiden Firmen wird scheinbar jetzt mit anderen Waffen fortgesetzt.

Schließlich wählt jeder Spieler aus einer vorgegebenen Tabelle eine ihm genehme Farbe (Farbnummer eingeben!) aus, und los geht's.

Verteilung der Länder

Was ein rechter Feldherr ist, der herrscht natürlich über eine Reihe von Ländern. Insgesamt ist die Spielwelt in 42 fiktive Staaten aufgeteilt. Um nicht gleich zu Beginn des Spiels einen Streit vom Zaun zu brechen, werden jedem Mitspieler rein zufällig Länder vom Programm zugeteilt. Die 42 Staaten werden also durch die Zahl der Mitspieler geteilt und die jedem zustehende Anzahl vergeben. Sie erkennen Länder eines Feldherren an der gleichen Farbe (Bild 1).

Jetzt startet das Spiel, in dem es für alle um das gleiche Spielziel geht: Erobern Sie die Welt.

Ein zufällig gewählter Spieler beginnt die Aktionen. Doch wie soll man mit Staaten andere Länder erobern? Natürlich benötigt man dazu Streitkräfte, die in unserem Spiel als »Armeen« bezeichnet werden. Jeder Spieler erhält für jedes Land anfangs eine Armee. Ist ein Spieler nun an der Reihe, erhält er nach einer einfachen Formel neue Armeen hinzu. Die Formel lautet:

```
Besetzte Länder /3
= neue Armeen
```

Das Minimum an zugeteilten neuen Armeen ist jedoch immer drei, auch wenn ein Spieler vielleicht nur noch ein Land besitzt.

Wir stehen nun am Anfang des Kampfes um die Spielewelt. Sechs Teilnehmer warten auf die ersten Schwächen ihrer Gegner. Der erste Spieler erhält drei äußerst muntere Armeen und steht vor der Frage, in welches seiner Länder er diese nun einquartiert. Die Farbe des Spielers erkennen Sie an der Figur unten rechts. Ist die Entscheidung getroffen, bewegt er den Mauszeiger auf dieses Land (es muß natürlich schon ihm gehören) und klickt es mit der linken Maustaste an. Der Spieler entscheidet weiterhin, wie viele seiner Armeen er in dieses Land positioniert. Eine beliebige Aufteilung der Streitkräfte ist regelgerecht, jedoch anfangs nicht sinnvoll.

Sind die verteilten Armeen einquartiert, ändert sich die Zahl in dem betreffenden Land. Sie zeigt nun die aktuelle Anzahl der Armeen.

Gut, jetzt hat unser Spieler seine Armeen in ein Land verfrachtet. Und jetzt? Alle Spieler haben ja die Aufgabe, die Welt mit den 42 Staaten zu erobern. Fangen wir also am besten damit an, ein neues Land zu erkämpfen.

Aktivieren Sie die Menüleiste mit der rechten Maustaste. Sie finden oben links auf dem Bildschirm das Menü »Aktionen«. Fahren Sie bei gedrückter rechter Maustaste auf dieses Menüfeld. Ein Fenster klappt auf, in dem Sie fünf Unterpunkte sehen:

Angriff

Wählen Sie bei gedrückter rechter Maustaste den Punkt Angriff und lassen die Taste los.

Im unteren Bereich des Spielfelds erscheint das Wörtchen »Von«. Bewegen Sie nun den Mauszeiger auf Ihr Land, von dem Sie den Angriff starten wollen. Klicken Sie dieses mit der linken Maustaste an. Sollte nur eine Armee in dem gewählten Land stehen, so meldet das Programm »Hier gibt es keine Angriffsarmeen!«. Wir lernen daraus, daß mindestens zwei Armeen in einem Land stehen müssen, von dem aus ein Angriff erfolgen soll. Haben Sie ein eigenes Land gewählt, in dem mehr als eine Armee steht, fragt das Programm nach dem Ziel Ihrer Wünsche, also nach dem anzugreifenden Land. Da wir es ja mit soliden Armeen zu tun haben, die nicht fliegen können, ist ein Land auszuwählen, das direkt benachbart ist. Dieses ist ebenfalls anzuklicken, und schon tobt eine Schlacht um das Zielland.

In Tabelle 1 finden Sie die Übergänge zwischen Ländern, die nicht direkt aus der Karte deutlich werden.

Die Entscheidung, welcher Feldherr zukünftig das angegriffene Land zu seinem Besitz zählen darf, fällt jetzt. Angenommen, Sie sind der Angreifer und haben vier Armeen im Basisland. Der Verteidiger besitzt nur eine einzige Armee in dem angegriffenen Land. Die Entscheidung über Sieg oder Niederlage in einer Schlacht fällt mittels simulierten Würfelergebnissen. Der Angreifer besitzt vier Armeen, drei davon sind für den Angriff einzusetzen. Die drei Angriffsarmeen werden in der Entscheidung durch drei Würfel symbolisiert. Die verteidigende Streitmacht des Gegners, eine Armee, stellt einen Würfel. Das Verhältnis bei diesem Angriffswurf lautet also 3:1. Nun wird gewürfelt, das höchste Einzelergebnis eines Wurfes entscheidet. Die Würfelergebnisse sind am rechten unteren Bildschirmrand in einem Kasten darge-

Land 1	Land 2
Alaska	Kamtschatka
Grönland	Island
Island	Skandinavien
Island	Großbritannien
Großbrit.	Skandinavien
Großbrit.	Westeuropa
Mittl. Osten	Ostafrika
Mittl. Osten	Ägypten
Südeuropa	Mittl. Osten
Brasilien	N/W-Afrika
Südeuropa	N/W-Afrika
Südafrika	Madagaskar
Indonesien	Siam
Siam	Neu-Guinea
Neu-Guinea	Indonesien
Neu-Guinea	Ost-Australien
Indonesien	West-Australien
Indonesien	Ost-Australien

Tabelle 1. Diese Länder sind auch Nachbarländer

Kontinent	Armeen
Asien	7
Nordamerika	5
Europa	5
Afrika	3
Südamerika	2
Australien	2

Tabelle 2. Diese Anzahl an Armeen gibt es zusätzlich für besetzt gehaltene Kontinente



Bild 1. Erobern Sie die Welt, eine farbenfrohe Karte dient als Spielfeld



Bild 2. Der Spieler »pc« ist an der Reihe, er besitzt die Länder mit der Farbe, die der Soldat unten rechts zeigt



Bild 3. Alea jacta est — Die Würfel sind gefallen. Wird der Angreifer zum erfolgreichsten Feldherren?



Bild 4. Symbole für die Karten sind Soldat, Reiter und Kanone — eine Serie bringt Zusatzarmeen

stellt. In unserem Fall zeigt dieser Kasten drei Ergebnisse für den Angreifer und einen Wurf für den Verteidiger. Die Ergebnisse werden jetzt miteinander verglichen. Hat der Angreifer als höchsten Wurf eine größere Punktezahl als der Gegner, zieht der Aggressor in das Land mit den drei Angriffsarmeen ein, die verteidigende Streitkraft wird vom Spielfeld entfernt. Bei Gleichheit der höchsten Punktzahl verliert der Angreifer, er ist also leicht im Nachteil. Hat der höchste Wurf des Angreifers weniger ergeben als der Würfel des Verteidigers Augen zeigt, verliert der Angreifer selbstverständlich auch. Haben Sie noch mehr als eine Armee im Ausgangsland, werden Sie gefragt, ob Sie den Angriff wiederholen wollen. Bitte klicken Sie dann das Feld »Ja« an.

Im späteren Spielverlauf kann es zu einer Ansammlung von Armeen kommen, die dann in gewaltigen Schlachten um Länder kämpfen. In einem Wurf kann jedoch maximal mit je drei Würfeln »gekämpft« werden. In dem Fall, daß der Angreifer und der Verteidiger mehr als einen Würfel (also eine Armee) einsetzen, gilt folgende Regel:

Die Würfel beider Kontrahenten werden nach der Höhe sortiert, und dann jeweils miteinander verglichen. Bei Gleichheit gewinnt wieder der Verteidiger. Ein Beispiel:
Wurf Angreifer: 6, 4, 1
Wurf Verteidiger: 6, 3, 2

Das Ergebnis sagt aus, daß der Angreifer zwei Armeen verliert und der Verteidiger eine. Die Sechsen waren gleich, der Verteidiger gewinnt. Die Vier des Angreifers war höher als die Drei des Verteidigers, eine Armee dieses Spielers muß vom Schlachtfeld. Die Zwei des Verteidigers übertrumpft wieder die Eins des Angreifers, damit verliert dieser seine zweite Armee. Setzt der Verteidiger nur eine Armee ein, verliert der Angreifer bei einem niedrigen Wurf immer höchstens eine Armee, bei zwei Verteidigern höchstens zwei.

Die Berechnungen übernimmt in unserem Spiel natürlich der Amiga.

War der Angriffszug erfolgreich und die letzte Armee des Gegners geschlagen, rücken die siegreichen Armeen in das Zielland ein. Jetzt kann der Spieler, der an der Reihe ist, entscheiden, ob er

- weiter angreift,
- Armeen verschiebt,
- Karten tauscht, oder
- dem nächsten Spieler die Aktionen überläßt.

Die Spielwelt ist in die Kontinente Nordamerika, Südamerika, Europa, Asien, Afrika und Australien unterteilt. Der Besitz eines Kontinentes über eine ganze Runde hinweg wird belohnt. Der Spieler erhält vor seinen Aktionen die normale Anzahl der neuen Armeen. Zusätzlich gibt es als Bonus für den gehaltenen Kontinent eine bestimmte Anzahl. Die Werte entnehmen Sie bitte Tabelle 2. Es ist also ratsam, schnell einen Kontinent zu erobern, um



DELUXE:
Software für den Amiga.

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

224/805

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526; Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

in den Genuß des Bonus zu kommen. Versuchen Sie den Kontinent möglichst lange zu halten, denn die Zusatzarmeen gibt es in jeder Runde erneut.

Es ist nach einiger Spielzeit unbedingt notwendig, Truppen zu verschieben. Ist beispielsweise ein Kontinent erobert, sind die Armeen im Inneren dieses Kontinents nicht mehr besonders effektiv. Also schiebt man die Truppen besser an die Grenzen. Dies geschieht mit dem Menüpunkt »Verschieben«. Die Bewegung ist immer nur einmal über eine Landesgrenze gestattet. Dabei können beliebig viele Armeen verschoben werden. Eine Verteidigungseinheit muß jedoch immer im Ausgangsland bleiben.

Ist ein weiterer Angriff mit diesen Armeen erfolgt, kann wieder verschoben werden. Der Sinn ist dabei, bei einem großen Feldzug immer wieder Nachschub heranzuziehen.

Haben Sie eine Funktion, zum Beispiel »Angriff«, versehentlich gewählt, können Sie das Feld »Abbruch« (Mitte unten) anklicken. Wählen Sie dann die nächste Operation. Der Abbruch ist beispielsweise auch dann notwendig, wenn Sie ein Land für einen Angriff ausgewählt haben, das nur



Bild 5. Hier legt der Spieler »napoleon« fest, wie er die Armeen verteilt, die er in dieser Runde erhält.

von eigenen Ländern umgeben ist. Sie erhalten dann immer die Meldung »Dies ist Dein eigenes Land ! Noch mal«.

Ein Feldherr sollte möglichst gut mit Karten umgehen können. Das gilt auch für unser Strategiespiel. Wurde mindestens ein Land in einer Spielrunde erobert, gibt es eine der wertvollen Karten. Die Karten tragen die Symbole Soldat, Reiter und Kanone (Bild 2). Besitzt ein Spieler drei Karten mit dem gleichen Symbol oder eine Serie aller Symbole, dann kann er diese gegen Armeen eintauschen. Der erste Spieler, der tauscht, erhält drei Armeen, der zweite sechs und dann jeweils drei Armeen mehr. Auf diese Weise ist nach mehrfachem Tauschen die Anzahl der Bonusarmeen schnell angestiegen. Umwälzende Veränderungen sind mit dem Einsatz der Zusatzarmeen schnell herbeigeführt.

Aus den genannten Gründen sollten Sie unbedingt versuchen, in jeder Spielrunde mindestens ein Land zu erobern.

Ein erfolgreicher Feldherr schnappt sich so schnell wie möglich einen der kleinen Kontinente. Diese sind leichter zu halten als die drei großen — und nur in diesem Fall gibt es den Bonus. Ist der Kontinent mit zahlreichen Verteidigern geschützt, setzen Sie Ihre Eroberung fort.

Tauschen Sie die erworbenen Karten nicht zu früh, warten Sie möglichst ab, bis sich der Tausch wirklich lohnt.

Kämpfen Sie immer nur so lange gegen einen starken Gegner, bis Ihre Armeen auf ein niedriges, aber verteidigungsfähiges Niveau geschrumpft sind. In jeder Runde sollten Sie einen Kampf gegen einen sehr schwachen Gegner führen, um eine Karte zu erhalten — die später zu Bonusarmeen führt.

Bei mehreren Mitspielern sind Bündnisse eine tolle Sache. Kämpfen Sie gemeinsam gegen einen Gegner, bewahren Sie den Frieden an gemeinsamen Grenzen. Wenn Sie von einem »Bündnispartner« überfallen werden, nehmen Sie das nicht persönlich: Es ist doch nur ein Spiel!

(Jörg Horn/Dietmar Körtke/rs)

Fortsetzung des Artikels auf Seite 134

Sagen Sie uns die Meinung!

Helfen Sie mit, Ihr Amiga-Sonderheft noch besser zu gestalten. Wenn Sie diesen Fragebogen ausfüllen und an unserer Leserumfrage teilnehmen, haben Sie zudem die Chance, ein Overdrive-Doppellaufwerk für Ihren Amiga zu gewinnen.

Wenn Sie sich zehn Minuten Zeit nehmen, haben Sie nicht nur die Chance, in Zukunft ein noch besseres Amiga-Sonderheft zu bekommen, sondern können auch ein Overdrive-System-Doppellaufwerk (Bild 1) der Firma Soyka Datentechnik in Bochum gewinnen. Diese Laufwerke zeichnen sich durch hohe Datensicherheit, geringes Geräuschniveau und sehr gute Abschirmung aus.

Durch Ihre Antworten auf unserem Fragebogen sagen Sie uns, was Ihnen gefällt, was besser werden soll und was Sie zukünftig im Amiga-Sonderheft lesen wollen.

Bevor Sie sich auf den Fragebogen stürzen, sollten Sie einige Dinge wissen:

— Ihre Antworten, ob Lob oder Kritik, haben keinen Einfluß auf Ihre Gewinnchancen
— Bei fast allen Fragen sind Mehrfachnennungen möglich
— Fragen, bei denen freie Textangaben vorgesehen sind (Typ des Druckers etc.), tragen Sie bitte eine möglichst genaue Typenbezeichnung ein. Zum Beispiel: »Drucker NEC P6 C«.
— Sie müssen Ihr Heft nicht zerschneiden, eine Fotokopie der Umfrage-Seiten genügt auch.
— Wenn Sie an der Verlosung des Overdrive-Systems teilnehmen wollen: Vergessen Sie Ihre Adresse nicht.

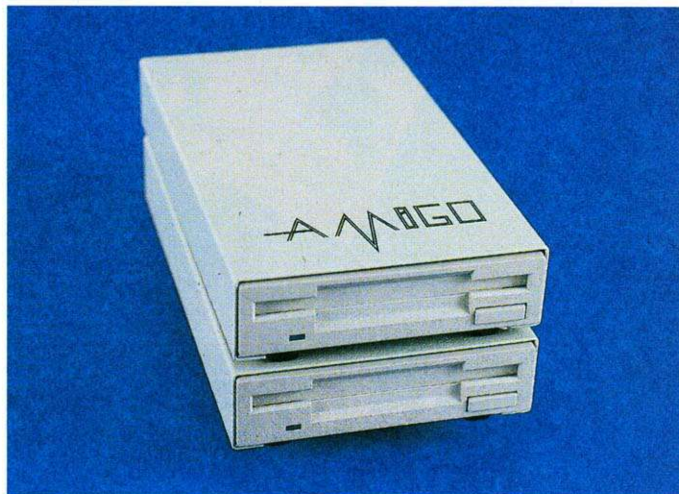


Bild 1. Eine tolle Sache, ein Doppellaufwerk am Amiga

Schicken Sie Ihren ausgefüllten Fragebogen an:

Markt & Technik Verlag AG
Redaktion Amiga-Sonderheft/Leserumfrage
Hans-Pinsel-Straße 2
8013 Haar bei München

Einsendeschluß ist der 31.12.1988 (Datum des Poststempels).

Der Rechtsweg ist ausgeschlossen. Mitarbeiter der Markt & Technik Verlag AG sowie deren Angehörige dürfen an der Umfrage nicht teilnehmen.

1. Welche Themenbereiche sollten in den Amiga-Sonderheften angesprochen werden.
(Bewerten Sie die Wichtigkeit mit einer Note zwischen 1 und 6)

Anwendungslistings	____
Bauanleitungen	____
Betriebssysteme	____
Bücher	____
CLI	____
DFÜ	____
Digitaliseren/Scannen	____
Drucker-Tests	____
Einsteigerthemen	____
Festplatten/Floppylaufwerke	____
Grafik	____
Grafik-Animation	____
Grundlagen	____
Hardware-Tests	____
Intuition	____
Kaufmänn. Anwendungen	____
Messen/Steuern/Regeln	____
Musik/Midi	____
Problemlösungen	____
Programmieren	____
Programmierkurse	
- Assembler	____
- C	____
- Basic	____
- Pascal	____
- Modula	____
Programmiersprachen allgemein	____
Schule/Ausbildung	____
Software-Tests zu einem bestimmten Thema	____
Spiele-Listings	____
Stories	____
Techn./Wiss. Anwendungen	____
Themen für Profis	____
Tips & Tricks	____
Video-Anwendungen	____
Workbench	____

2. Wie ist Ihre Meinung zum Amiga-Sonderheft
Bewerten Sie die einzelnen Punkte mit einer Note zwischen 1 und 6)

Informativ	____
Aktuell	____
Kritisch	____
Sachlich	____
Verständlich	____
Ansprechend	____
Preiswert	____
Unverzichtbar	____

3. Besitzen Sie einen Amiga?
Nein O Amiga 500 O Amiga 1000 O Amiga 2000 O

4. Wollen Sie demnächst einen Amiga kaufen?
Nein O Amiga 500 O Amiga 1000 O
Amiga 2000 O Amiga 2500 UX O Amiga 2500 AT O

5. Besaßen Sie oder besitzen Sie einen anderen Computer?
Nein O Ja O
Wenn ja, welchen?

	Wird noch benutzt	Besitze ich	Besaß ich
C 64	O	O	O
CPC	O	O	O
Atari XL/XE	O	O	O
Atari ST	O	O	O
PC	O	O	O

Verkauft am: ______

Andere: ____________________

6. Wie lange besitzen Sie schon Ihren Amiga?
Weniger als drei Monate O
Bis sechs Monate O
Bis ein Jahr O
Über ein Jahr O
Über zwei Jahre O

7. Wie ist Ihr Amiga ausgestattet?
a) Monitor
Spezieller Amiga-Monitor O Fernsehgerät O
Multtisync O
b) Massenspeicher
Diskettenlaufwerke O Anzahl ____
Festplatte O Anzahl ____ Kapazität ____ MByte
Sidecar O
PC-Karte (A 2000) O

c) Speichererweiterungen
keine O 512 KByte O
1 MByte O 2 MByte O
3 MByte O Mehr als 3 MByte O

d) Drucker
Keinen O Matrix O Laser O
Tintenstrahl O Thermo O
Typ: ____________________
Farbdrucker: Ja O Nein O

e) DFÜ
Akustik-Koppler O Modem O

f) Digitizer/Scanner/Genlock etc.
Typ: ____________________

8. Wie wird Ihr Amiga von Ihnen vorwiegend genutzt?
Im privaten Bereich:
Programmieren O Anwendungen O
Spielen O

In der Ausbildung:
Schule O Fortbildung O
Studium O

Im Beruf:
Hauptberuflich: O
Nebenbeschäftigung O

UMFRAGE

9. Was machen Sie mit Ihrem Amiga

	intensiv	gelegentlich	nie	würde mich interessieren
Animation	O	O	O	O
Dateiverwaltung	O	O	O	O
DFÜ	O	O	O	O
Desktop Publishing	O	O	O	O
Digitalisieren/Scannen	O	O	O	O
Elektronik/Basteln	O	O	O	O
Grafik/Malen	O	O	O	O
Kaufm. Anwendung	O	O	O	O
Lernprogramme	O	O	O	O
Messen, Steuern, Regeln	O	O	O	O
Musik/Midi	O	O	O	O
Programmieren	O	O	O	O
Spielen	O	O	O	O
Textverarbeitung	O	O	O	O

Sonstiges

____

10. Wie stufen Sie sich ein?

Einsteiger ohne Vorkentnisse O Einsteiger O Fortgeschrittener O
Erfahrener Fortgeschrittener O Profi O Ist mein Beruf O

11. Was machen Sie mit den Listings im Amiga-Sonderheft?

Tippe alle ein O
Tippe nur die kurzen ein O
Interessiere mich nur für die Struktur O
Interessieren mich nicht O
Besorge mir sie auf Diskette von Freunden O
Nehme den Programmservice in Anspruch O

12. Wie groß sollte Ihrer Meinung nach der Listing-Teil im Amiga-Sonderheft sein?

ca. 20 Seiten O ca. 40 Seiten O
ca 50 Seiten O mehr als 50 Seiten O

13. Wie viele Listings sollten in einem Amiga-Sonderheft abgedruckt werden?

Bis 5 Listings O Bis 10 Listings O
Bis 15 Listings O Mehr als 15 Listings O

14. Wie stehen Sie zu der Möglichkeit, Programme nicht mehr im Heft abzudrucken und diese nur noch auf Diskette anzubieten. Die Information und die Zahl der Programme im Amiga-Sonderheft könnte so wesentlich höher sein

Alle Programme nur auf Diskette O
Nur besonders lange Programme nur auf Diskette O
Alle Programme auch im Amiga-Sonderheft als Listing O

15. Welche Programmiersprachen kennen Sie, interessieren Sie?

Sprache	kenne ich	besitze ich	beherrsche ich	würde ich gerne im Heft haben	wünsche ich ich mir einen Kurs
Assembler	O	O	O	O	O
ADA	O	O	O	O	O
Basic	O	O	O	O	O
C	O	O	O	O	O
Forth	O	O	O	O	O
Logo	O	O	O	O	O
Lisp	O	O	O	O	O
Modula	O	O	O	O	O
Pascal	O	O	O	O	O
Prolog	O	O	O	O	O

Andere______________________________

16. Fragen zur Person

a) Alter: ________ Jahre
b) Geschlecht männlich O weiblich O

c) Welche Hobbies haben Sie?

Basteln O
Computer O
Elektronik O
Freizeit-Verein O
Sport O
Lesen O
Sonstige O

d) Schulabschluß, Ausbildung, bzw. angestrebter Abschluß

Hauptschule O Mittlere Reife O Lehre O
Fachoberschule O Abitur O
Fachhochschule O Universität O

e) Anschrift
(Ausfüllen, wenn Sie an der Verlosung teilnehmen wollen)

Name:
____

Straße:
____

Ort:
____

Telefon:
____

Ich bin damit einverstanden, daß meine hier gemachten Angaben elektronisch ausgewertet werden.

Unterschrift

____

Profilaufwerk 3,5"
Metallgehäuse ● einstellbare Lauwerknummer mit Displayanzeige ● digitale Trackanzeige ● Write Protect am Laufwerk schaltbar ● abschaltbar ● durchgeschleifter Bus
1 Jahr Garantie

Super ALCOMPreis	329,-

Laufwerk 5,25"
40/80 Track ● Laufwerksbus durchgeschleift ● abschaltbar ● einstellbare Adressen ● MS- DOS- kompatibel ● mit Diskchange

Super ALCOMPreis	298,-
HD 1,6 MB (umschaltbar)	318,-
Amigafarbene Blende	+10,-
Write Protect Schalter	+15,-

Gemischtes Doppel 3,5/5,25"
einzeln ein-/abschaltbar ● einstellbare Laufwerksnummern mit Anzeige ● durchgeschleifter Bus ● bei 5,25" 40/80 Tracks umschaltbar ● Metallgehäuse ● 1 Jahr Garantie

Super ALCOMPreis	548,-

500er Speichererweiterung
Für 512k zusätzliches RAM ● alle RAM-s gesockelt ● selbstkonfigurierend ● abschaltbar ● Uhrenschaltung auf Platine mit Akku- bzw. Batteriepufferung nachrüstbar

Komplett mit 512k	Preis auf Anfrage
Superpreis mit Uhr	Preis auf Anfrage
Bauteilesatz für Uhr ohne Akku	24,-
Leerplatine mit Stecker	*39,-

*mit Schaltplan und Bestückungsliste

Laufwerkanschlußkabel
Zum Anschluß von Laufwerken an alle Amigas ● mit Ansteuerelektronik

Für 3,5" Laufwerk	39,-
Für 5,25" Laufwerk	49,-

Steckplatzerweiterung 3-fach für Laufwerke
Jeder Steckplatz abschaltbar und einstellbare Laufwerksnummer ● Steckplatzerweiterung direkt am Amigagehäuse ● Dadurch keine Kabellängenprobleme

Anschlußfertig zum Super ALCOMPreis	49,-

Soundsampler
Für alle Amiga's mit Software ● Type bei Bestellung bitte angeben ● 8-Bit Datenbreite ● Betrieb am Parallelport (Druckerport) ● Mit Vorverstärker für Micro-Anschluß (Cinch-Buchsen) ● Musik- und Sprachdigitalisierung möglich ● Arbeitet mit fast allen Digitizer-Programmen ● Formschönes Gehäuse

Super ALCOMPreis	79,-

Sampler Studio
● Professionelles Sampler-Programm ● 4-Kanal-Technik ● speichern auf 4 Disketten hintereinander möglich ● alle gängigen Formate (IFF, Data, Future) ● Echtzeitdisplay mit Zoomfunktion ● viele Verfremdungsmöglichkeiten ● Echo, Hall, Reverse

	69,-
Paket: Sampler + Software	129,-

MIDI - Interface
4 Kanäle einschließlich 1 Thru ● Optische Datenanzeige ● Formschönes Gehäuse

Wahnsinnspreis von nur	89,-

Frischer Wind auf dem Hardwaresektor

ALCOMP
COMPUTERHARDWARE

Kickstartumschaltung
Bauen Sie die anderen Kickstart-Versionen in Ihren Amiga 500 ● Einfacher Einbau ohne Löten ● für Original-Kickstart-ROM und 2 zusätzliche Versionen auf EPROM ● EPROM-Programmierservice auf Anfrage

SuperALCOMPreis	59.-
Kickstartversion auf EPROM's	120.-

Userport + Experimentierkarte für Expansionport
Mit Lochraster und 2 x 6522 Ports

Leer	59.-
komplett aufgebaut	89.-

Wir suchen ständig Hardware-Entwicklungen. Wir garantieren gute Umsatzprovisionen und ehrliche Abrechnung

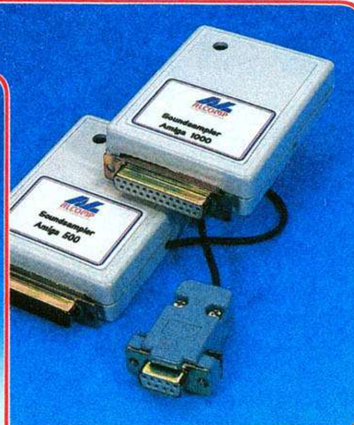
ausgereifte Ingenieurleistung ● 14 Tage Umtauschrecht ● fast alle IC's gesockelt ● nur professionelle Leiterplatten ● Bauteile namhafter Hersteller ● mit Bedienungsanleitung



kostenloses Info anfordern!!!

Bestellung und Versand
ALCOMP
A. Lanfermann
Lessing Str. 46
5012 Bedburg
Tel. 0 22 72/15 80

Nachnahmeversand NN-Spesen 7.50 DM b. Vorkasse 3.- DM. Auslandsbestellungen: Nachnahmeversand NN-Spesen 10.- DM b. Vorkasse 5.-DM. Wir liefern Ihnen auf Ihre Rechnung und Gefahr zu den Verkaufs- und Lieferbedingungen des Elektronikgewerbes.
Postgiroamt Köln
(BLZ 370 100 50) 275 54-509



3,5" Laufwerk
Für alle Amiga's ● einstellbare Gerätenummer ● abschaltbar ● Metallgehäuse ● superflach ● 1 Zoll (2,54cm) ● durchgeschleifter Bus ● TEAC Laufwerk
1 Jahr Garantie

komplett anschlußfertig	239,-
Amigafarbene Blende	+10,-



Basislaufwerke
1 Jahr Garantie

TEAC FD 135 FN 3,5" 1MB superslimline	218,-
TEAC FD 55 GFR 5,25" 40/80 Tracks	239,-
Amigafarbene Blende	+10,-
1,6 MB Diskchange	259,-
3,5" Gehäuse	25,-
5,25" Gehäuse	25,-
Gehäuse für "Gemischtes Doppel"	65,-

Bootselector 19,90

Amiga Eprommer
- Für A 500/1000
- Expansionsportanschluß
- Für EPROM's 2764-27011 (8K-128K) Alle A-Typen und CMOS-Typen
- Funktionen:
  LEERTEST LADEN VON DISK
  VERGLEICHEN SPEICHERN AUS DISK
  AUSLESEN HEXDUMP
  BRENNEN
- vier Programmieralgorithmen 50mS/Byte - Superschnell 64K-1,5 min
- Programm zum Generieren und Brennen von Kickstarts direkt von Diskette oder aus ROM
- Mit Software + Gehäuse **225,-**

Meß- und Steuerinterface
- 8 ADC-Kanäle 0-2,55V in 0,01V Stufe
- 1 DAC-Kanäle 0-2,55V in 0,01V Stufe Genauigkeit- 1,5 LSB
- 8 frei programmierbare TTL-I/O Kanäle
- Mit Gehäuse, Anschlüße auf Schraubklemmen
- interne Referenzspannung
- Expansionsanschluß
- Einfache Programmierung in Basic möglich Multitasking tauglich
- incl. DEMO-Software auf 3,5" Diskette

239,-

Trackanzeige
Für DFO-DF3 einstellbar ● für alle Laufwerke (3,5"/5,25") ● Laufwerkbus durchgeschleift ● mit Gehäuse

Super ALCOMPreis	69,-

Einführungsangebot Amiga - Harddisks

HD-Interface A 2000	198,-
HD-Interface A 500/A 1000	249,-
Platte 20 MB A 2000	798,-
30 MB A 2000	898,-
40 MB A 2000	1 098,-
65 MB A 2000	1 348,-
Platte A 500/A 1000 20 MB	898,-
30 MB	998,-
40 MB	1 248,-
65 MB	1 498,-

Vokabeltrainer
2500 englisch-deutsche Vokabeln incl. Hilfssatz ● Merkfunktion ● komfortabler Editor zur Vokabelverwaltung ● Wörterbuch zum Dateiendurchsuchen

59,-



Selbstbootende Harddisk für Amiga ohne PC-Karte!
Die Amiga-Festplatte von ALCOMP:
● Selbstbootend wie "Card" oder "Rad"! ● Als Einbau-Festplatte für den "Amiga 2000" ● Als Externe Einheit für den "Amiga 500" und 1000 mit Gehäuse, eigenem Netzteil und Erweiterungsanschluß ● Erhältlich mit 20, 30, 40 und 65 Megabyte ● Kopiert 1 Megabyte in unter 4 Sekunden ● Speichert schneller als "1.2-Ramdisk" ● Läuft mit "FastFileSystem" ● Einfach einstecken, Formatieren, "Mountlist" und "Startup-Sequence" ändern und los geht's!
Entwickler: Stephan und Stefan
Für den Selbstbau: Harddisk-Interface incl. Steuersoftware ● Anschluß mit Slot für Omti-Controller

Entwurf + Druck PLUM, Düren-Oberzier, TEL. 0 24 28/7 11

Finanzen im Griff



Überlassen Sie nichts dem Zufall. Wenn es um Ihr Geld geht, genügt kein gelegentlicher Blick auf den Bankauszug von letzter Woche. Mit dem elektronischen Haushaltsbuch haben Sie Ihre Kasse stets unter Kontrolle.





Mit dem »Haushaltsbuch« für den Amiga wird die Überwachung Ihrer täglichen, monatlichen oder jährlichen Einnahmen und Ausgaben zum Kinderspiel. Die Bedienung des Programms ist schnell erlernt, die Menüs sind mit der Maus aufzurufen. Lediglich die Eingabe der Kontennamen und der Geldbeträge erfolgt über die Tastatur.

Haben Sie erst einmal ein Konto angelegt, die ersten Beträge verbucht und den Stand auf Diskette gespeichert, genügt ein Mausklick, um eine Monats- oder Jahresübersicht auszugeben. Weitere Menüs wie DISK, ZEIT, SYSTEM, AUSGABE und DRUCKER geben dem Programm einen professionellen Touch.

So führen Sie Ihre Konten

Der Arbeitsbildschirm des Basic-Programms »HAUSHALTSBUCH« (Listing 1) unterteilt sich in zwei Windows:

1. Das Eingabe- und Ausgabefeld (oben).

2. Aktuelle Informationen wie Uhrzeit, Speicherplatz und Programmhilfen.

Vor dem ersten Start des Programms sind einige Vorarbeiten nötig. Beachten Sie bitte den Kasten mit den Hinweisen zu diesen Punkten. Besitzer der Programmservice-Diskette können allerdings sofort mit der Verwaltung beginnen.

Die Programmdiskette muß sich im Laufwerk df0: befinden. Basic-Programmierer können das Programm bei Bedarf jedoch leicht umschreiben, so daß ein zweites Laufwerk angesprochen wird.

Menü à la carte

Das Haushaltsbuch stellt zahlreiche Funktionen zur Verfügung, die bei gedrückter rechter Maustaste (Menütaste) angewählt werden.

Bevor Sie mit der Eingabe von Daten beginnen, ist ein neues Konto anzulegen. Insgesamt können zehn Konten in einer Datei geführt werden, was für die private Anwendung sicher ausreicht.

Wählen Sie den Punkt »neues Konto« im Menü »Option« (Bild 1). Geben Sie anschließend den Namen, beispielsweise »Gehalt«, ein. Speichern Sie diesen auf Diskette, indem Sie die folgende Frage mit »Ja« beantworten (mit der Maus anklicken).

Wenn Sie anschließend im »Verzeichnis« nachschauen, finden Sie dort den Eintrag des angelegten Kontos.

Die erste Buchung

Sie können nun weitere Konten eröffnen, oder aber in das eröffnete Konto Einträge buchen. Wählen Sie dazu mit der Maus im »Verzeichnis« das Konto, in das Einträge gebucht werden sollen. Sie erkennen die erfolgreiche Auswahl daran, daß im Ausgabefeld unten links der Kontoname erscheint.

Öffnen Sie dann das Menü »Option« und wählen »Buchen«. Bestimmen Sie anschließend, ob eine Ausgabe oder eine Einnahme verbucht werden soll. Sie werden dann aufgefordert, eine Bezeichnung für den Buchungsbetrag einzutippen. Schließen Sie die Eingabe mit <RETURN> ab. Danach ist die Höhe des Betrags einzugeben. Hier sind nur Zahlen und, als Trennung für die Nachkommastelle, der Punkt erlaubt.

Sollten Sie versehentlich ein ungültiges Zeichen eingegeben haben, erscheint die Meldung »?Redo from start«. Drücken Sie dann <RETURN> und buchen erneut.

Die in der Arbeitssitzung bisher erfolgten Buchungen sehen Sie mit der Wahl des Menüpunktes »Buchungen«.



Haben Sie auf die beschriebene Art alle Buchungen vorgenommen, speichern Sie diese mit dem Punkt »Eintragen« auf Diskette. Erst dann sind Ihre Daten endgültig festgehalten. Bei längeren Sitzungen sollten Sie zwischendurch immer wieder auf Diskette speichern, um Datenverlust, beispielsweise durch einen Stromausfall, zu vermeiden.

Sie können die auf Diskette gespeicherten Daten in die-

ANWENDUNGEN















```
von A.N. u. M.F.          !!!Haushaltsverwaltung!!!          c 1988

Bitte neuen Kontonamen eingeben        ? urlaubskosten_____

Zeit          12:23:28     Arbeitsdauer   h: 0 m: 0 s: 0
Datum         16-09-1988   Buchungen      0
Bezugszeit    16-09-1988   Speicher       530136  4407 26292
Kontoname
```

Bild 1. Legen Sie bis zu zehn Konten gleichzeitig an, die Auswahl erfolgt über diesen Menüpunkt

```
von A.N. u. M.F.          !!!Haushaltsverwaltung!!!
Einzelkonto  LEBENSMITTEL                 Jahr 1988 Monat09     Maustaste

Tag    Monat    Jahr    Bezeichnung           Betrag

16     09       1988    GROBEINKAUF          -455.78
16     09       1988    BRÖTCHEN               -2.80
16     09       1988    FLEISCH               -21.70
16     09       1988    SÜSSIGKEITEN           -4.55
                                 Saldo:      -484.83

Zeit          12:23:28     Arbeitsdauer   h: 0 m: 0 s: 0
Datum         16-09-1988   Buchungen      0
Bezugszeit    16-09-1988   Speicher       553632  4147 25722
Kontoname     LEBENSMITTEL
```

Bild 2. Die Buchungen eines Kontos sind mit dem Befehl »Übersicht Einzelkonto« darzustellen

```
von A.N. u. M.F.          !!!Haushaltsverwaltung!!!
Monatskonto            Jahr 1988 Monat 09             Maustaste

Konto                  Betrag

GEHALT                 15100.55
AUTO                       0.00
HOBBYS                     0.00
LEBENSMITTEL            -484.83
           Saldo:      14615.72

Zeit          12:42:14     Arbeitsdauer   h: 0 m: 0 s: 0
Datum         16-09-1988   Buchungen      0
Bezugszeit    16-09-1988   Speicher       556280  4407 26234
Kontoname     LEBENSMITTEL
```

Bild 3. Die Monatsübersicht zeigt Ihnen, wo in Zukunft gespart werden muß

sem Konto mit der Wahl »Übersicht Einzelkonto« einsehen (Bild 2). Die Funktionen »Übersicht Monat« und »Übersicht Jahr« zeigen alle gebuchten Einträge in dem betreffenden Zeitabschnitt (Bild 3). Die Ausgabe erfolgt dabei immer gemäß der eingestellten Bezugszeit.

Ein nicht mehr benötigtes Konto können Sie mit dem Punkt »Konto löschen« eliminieren. Es erfolgt vorher eine Sicherheitsabfrage.

Folgende Menüpunkte stehen weiterhin zur Wahl:

1. DISK

Hier können Sie das Directory einsehen, Files löschen oder einen Diskwechsel veranlassen.

2. ZEIT

Um einmal nachzuschauen, was in den vorangegangenen Monaten oder Jahren verbucht wurde, wird die Bezugszeit geändert. Wählen Sie den Punkt »Bezugszeit« und stellen mit der Maus die gewünschte Zeit ein (Bild 4). Bewegen Sie den Mauszeiger auf das gewünschte Feld. Drücken der linken Maustaste verändert die angezeigte Zeit. In der oberen Hälfte erhöht sich die Zeit, in der unteren erniedrigt sie sich. Sind Sie mit der Einstellung zufrieden, klicken Sie bitte das Feld »fertig« an. Die niedrigste Bezugszeit ist das Jahr 1988. Das Anklicken der Funktion »Systemzeit« genügt, um später zum aktuellen Datum zurückzukehren.

3. SYSTEM

Wählt man hier den Punkt »Farben«, können acht Farben mit drei Schiebereglern (Gadgets) verändert werden (Bild 5). Klicken Sie die zu verändernde Farbe an und stellen diese nach eigenem Geschmack ein. Auch diese Operation verlassen Sie durch Anklicken des Feldes »Fertig«.

Die Grundeinstellung erhält man mit der Funktion »Farbreset« zurück.

Der letzte Punkt im Menü »System« bewirkt das Programmende. Sollten sich noch gebuchte Einträge im Speicher befinden, werden Sie aufgefordert, diese vor dem Verlassen des »Haushaltsbuch« auf Diskette zu speichern.

4. AUSGABE

Hier wählen Sie die Statistik, die auf dem Drucker ausgegeben werden soll. Sie haben die Wahl zwischen Einzel-, Monats- oder Jahreskonto.

5. DRUCKER

In diesem Menü wählen Sie die Einstellungen für den angeschlossenen Drucker. Die Anpassung unterstützt Epson- und kompatible Drucker.

Das Programm »Haushaltsbuch« und der zugehörige Datenspeicher benötigen sehr

CLImate 1.2

Mit CLImate stehen Ihnen nun die Funktionen Ihres Amiga-Command-Line-Interface per Mausklick zur Verfügung

3½"-Diskette für die Amiga 500, 1000 und 2000

SOFTWARE

AMIGA

HiSoft Devpac Assembler deutsch

3½"-Diskette

HiSoft

SOFTWARE

AMIGA

ZING! ™ deutsch CLI-Tool

Bedienen Sie Ihr Amiga-Betriebssystem schnell und mühelos mit der Maus.

3½"-Diskette

MERIDIAN SOFTWARE INC.

SOFTWARE

AMIGA

ZING! KEYS ™ deutsch TOOL

MERIDIAN SOFTWARE INC.

Stoff für Ihren Amiga



Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik

Zeitschriften · Bücher

Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526; Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

227/808

viel Speicherplatz, so daß mindestens eine 512 KByte Speichererweiterung benötigt wird. Ist zusätzlich eine batteriegepufferte Echtzeituhr vorhanden, wird diese vom Programm unterstützt. Sollte die Uhr nicht vorhanden sein, müssen Datum und Uhrzeit vor dem Laden eingestellt werden.

(M. Fernholz/Walter Kurt/rs)

Jahr 1988 Monat 11 Tag 19 fertig

Zeit 12:42:14 Arbeitsdauer h: 0 m: 0 s: 0
Datum 16-09-1988 Buchungen 0
Bezugszeit 16-09-1988 Speicher 556224 4371 26174
Kontoname LEBENSMITTEL

Bild 4. Verändern Sie die Bezugszeit, um Buchungen ein anderes Datum als das Systemdatum zuzuweisen

von A.N. u. M.F. +++Haushaltsverwaltung+++ c 1988
Farben
0 1 2 3 4 5 6 7
rot
grün
blau

Zeit 12:42:14 Arbeitsdauer h: 0 m: 0 s: 0
Datum 16-09-1988 Buchungen 0
Bezugszeit 19-11-1988 Speicher 556160 4331 26148
Kontoname

Bild 5. Sogar die Farben des Arbeitsbildschirmes sind zu verändern

Setzen Sie bitte vor dem Start des Programms die Systemzeit, falls diese nicht mit dem aktuellen Datum und der Uhrzeit übereinstimmt. Diese Einstellungen führen Sie bitte mit der Workbench durch. Fortgeschrittene setzen die Zeit natürlich vom CLI aus.

Das Programm erwartet im Unterverzeichnis »SCHUBLADE« die vom Anwender eingestellte Farbtabelle. Da diese vor dem ersten Start des Programms nicht vorhanden sind, gehen Sie bitte folgendermaßen vor:

Legen Sie auf der Diskette mit dem Programm »Haushaltsbuch« das Subdirectory »SCHUBLADE« an. Dazu kopieren Sie das Verzeichnis »EMPTY« von der Workbench und benennen dies anschließend mit dem Rename-Befehl in »SCHUBLADE« um.

Legen Sie die Diskette in das Laufwerk df0:, laden Sie anschließend Amiga-Basic und dann das »Haushaltsbuch« (»OPEN« mit der rechten Maustaste anwählen). Geben Sie bitte im Basic-Fenster (links) nach Anklicken mit der linken Maustaste folgende Zeilen ein:

```
chdir "df0:schublade"
open "o",#1,"Zeit"
print#1,"195501"
close
```

Beachten Sie bitte unbedingt die korrekte Groß-/Kleinschreibung! Schreiben Sie nun im List-Fenster (rechts) vor die Zeile:

```
OPEN "A",#1,"Farben ...
```

die Buchstaben »REM«. Sie müssen den Mauszeiger dazu an den Anfang der Zeile bewegen und die linke Maustaste drücken. Ein senkrechter Strich erscheint in der Zeile. Geben Sie nun die Buchstaben »REM« ein, drücken einmal die Space-Taste und bewegen den Mauszeiger in die nächste Zeile. Starten Sie anschließend das Programm (»Start« anklicken). Mit der Option »Farben« im Menü »System« stellen Sie nun die Farben Ihrer Wahl ein. Wenn Sie die Farben nicht verändern wollen, klicken Sie bitte alle acht Farben der Reihe nach an und anschließend das Feld »Fertig«. Erst dann sind die Farbwerte zum Speichern vorbereitet!

Nach dem Einstellen fragt das Programm, ob Sie die Farben speichern wollen. Geben Sie unbedingt »JA« an. Das File »Farben« wird nun erzeugt. Beenden Sie anschließend die Arbeit mit dem Haushaltsbuch, erscheint die Meldung »Current program is not saved. Do you want to save it before proceeding?«. Klicken Sie mit der linken Maustaste bitte »NO« an. Nun kann das »Haushaltsbuch« ohne Schwierigkeiten gestartet werden.

Programmname:	Haushaltsbuch
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2
Bemerkung:	Bitte vor Programmstart im Unterverzeichnis »Schublade« die Dateien »Zeit« und »Farben« anlegen

```
 1 XbO start:
 2 Lc  REM +++Haushaltvervaltung von Andreas Noch und Martin Fernh
       olz +++
 3 gS  REM +++Erstellungsjahr Januar bis Februar 1988 +++
 4 He  REM  *** Bitte nur in einem Monat Buchen (EINTRAGEN!!!)
 5 QO  REM  *** Das Erstellen eines neuen Buchungsjahres dauert ca
       .1.5 min (Schreiben der Jahresdateien)
 6 NS  REM  *** Die Schublade MUß beinhalten die Dateien Zeit und
       Farben !!!!
 7 ZB  REM  *** Das Programm arbeitet fehlerfrei mit 1 Mbyte Speic
       her !
 8 L3  REM  *** Das Programm arbeitet auch mit 2 Laufwerken (Änder
       ung auf 2 Laufwerke CHDIRďf0:Schubladè Befehle im Program
       m auf DF1 verändern)
 9 zd  REM  *** Die niedrigste Bezugszeit ist 1988
10 Gj  CHDIR "df0:Schublade"
11 mK  CLEAR ,50000&
12 oD  OPTION BASE 1
13 5U  DEFDBL a-Z
14 XE  index=0
15 n5  DIM verzeichnis$(19),flag%(19),nummer(99),konto$(99),jahr$(
       99),monat$(99),tag$(99),buchung$(99),betrag$(99)
16 KS  GOSUB systemzeit
17 9o  GOSUB bildschirm
18 Mh  GOSUB einlesen
19 1i  OPEN "I",#1,"Farben":FOR i=0 TO 7:INPUT #1,rot(i+1),gruen
       (i+1),blau(i+1):PALETTE i,rot(i+1),gruen(i+1),blau(i+1):NEXT
       i:CLOSE #1
```

Listing 1. Das Haushaltsbuch bringt Ordnung in Ihre Finanzen. Bitte mit dem Checksummer (Seite 159) eingeben.

Fortsetzung des Listings auf Seite 143

Intelligenter Sprachentrainer

Ohne ausreichenden Wortschatz ist keine Fremdsprache zu beherrschen. Erweitern Sie Ihr Vokabular einer beliebigen Sprache mit diesem großartigen Programm. Mit dem Vokabel-Trainer sind nicht nur exotische Sonderzeichen darzustellen, er reagiert auch auf vorhandene Lücken in Ihrem Sprachschatz.

Die Fähigkeiten des »Vokabel-Trainers« (Listing 1 und 2) gehen weit über die Möglichkeiten vergleichbarer Programme hinaus. Eine Übersicht der hilfreichen Funktionen wird Sie überzeugen:

— mausunterstützte Benutzeroberfläche
— nationale Sonderzeichen auf Funktionstasten
— Suchfunktion für jede Vokabel oder das passende Synonym im Speicher und auf Diskette
— Vokabelabfrage nach drei verschiedenen Kriterien (Fremdwort, Übersetzung und Schreibweise)
— durch Statistikfunktion für jede einzelne Vokabel ständiger Überblick des eigenen Wissensstandes
— Datumsspeicherung der Abfrage

Fragen Sie sich, warum Vokabeln mit Hilfe eines Computers besser und schneller gelernt werden? Computerunterstütztes Lernen bietet eine Reihe von Vorteilen. Einerseits eignen Sie sich beim Eintippen wesentlich besser die Schreibweise der Vokabeln an. Andererseits lernen Sie die tatsächliche Bedeutungen einzelner Vokabeln und nicht eine Abfolge verschiedener Wörter. Beim üblichen Lernvorgang mit Hilfe eines Lehrbuches prägen sich die Vokabeln primär in der Reihenfolge der Lektion ein, weniger die Bedeutungen der einzelnen Vokabeln. Dies führt häufig zu dem Irrglauben, die Vokabeln seien schon bestens bekannt, obwohl bei einem aus dem Zusammenhang gerissenen Wort Unsicherheit auftritt.

Der Vokabeltrainer bietet weiterhin den großen Vorteil, Vokabeln dem Wissensstand des Benutzers angepaßt abzufragen. Dies geschieht durch ein automatisches Auswahlverfahren. Vokabeln, die Sie weniger gut beherrschen, werden häufiger abgefragt als bekannte Worte.

Die Möglichkeiten des Vokabeltrainers sind damit noch lange nicht erschöpft. Mehrere Anwender können ihre Vokabeln auf einer Diskette getrennt voneinander verwalten. Die Statistikfunktion erlaubt eine genaue Auswertung jeder einzelnen Vokabel, jeweils aufgeteilt auf die drei Abfragearten (siehe unten). Zusätzlich werden Datum und Uhrzeit mit auf Diskette gespeichert. Damit ist genau nachzuvollziehen, wann welche Vokabeln zuletzt abgefragt wurden und ob eventuell eine Wiederholung der entsprechenden Lektion nötig ist.

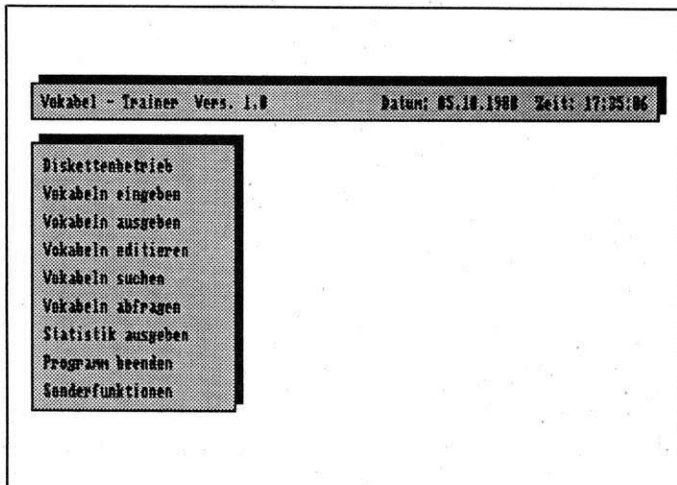


Bild 1. Das Hauptmenü des Vokabeltrainers

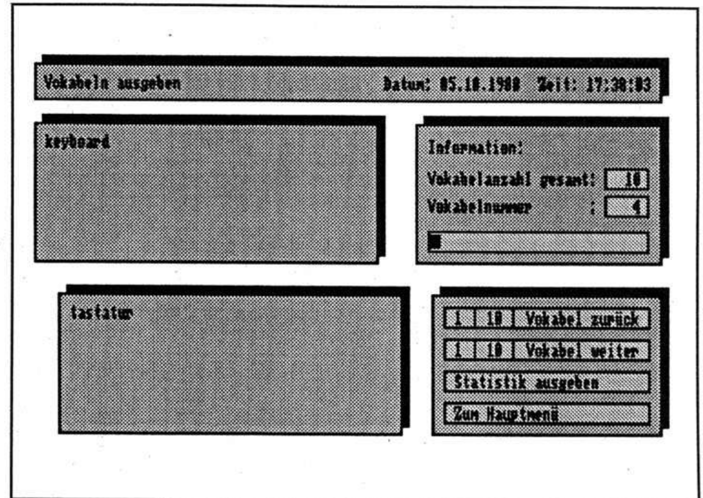


Bild 3. Die Maske für die Eingabe neuer Vokabeln

Die Vorbereitung

Im folgenden möchte ich zunächst einige Hinweise zum Abtippen geben und danach auf die einzelnen Funktionen des Programms ausführlich eingehen.

Der Vokabeltrainer ist vollständig in Amiga-Basic auf einem Amiga 2000 programmiert worden. Das Programm ist für »Basic-Kundige« leicht eigenen Wünschen anzupassen. Der Trainer läuft problemlos auf einem Amiga 500 oder Amiga 1000 mit einem Hauptspeicher von mindestens 512 KByte.

Vor der Eingabe des Hauptprogramms muß in jedem Fall der von Amiga-Basic zur Verfügung gestellte freie Speicherplatz von 25000 Byte auf mindestens 70000 Byte vergrößert werden. Für Einzeldateien mit mehr als 200 Vokabeln sollte er sogar etwa 100000 Byte betragen.

Das Hauptprogramm ist länger als 25000 Byte. Damit bei einem neu gestarteten Computer keine Schwierigkeiten auftreten, ist zunächst ein kleines Ladeprogramm mit Namen »Trainer« (Listing 1) zu laden und zu starten. Dieses Programm vergrößert den freien Speicherplatz und ruft das Hauptprogramm auf. Damit dieses Ladeprogramm das Hauptprogramm auf Diskette findet, speichern Sie den Hauptteil unter dem Dateinamen »Trainer HP« ab. Beachten Sie bitte, daß die Diskette sich im Laufwerk befindet, das im Basic-Lader angegeben ist (z.B. df0:).

Die maximal mögliche Anzahl von Vokabeln in einer Datei läßt sich leicht verändern, indem Sie die Variable »mmv« am Anfang des Programms Ihren Bedürfnissen anpassen. Ferner ist zu beachten, daß auch der im Ladeprogramm enthaltene Befehl »CLEAR« an den größeren Speicherbedarf angepaßt werden muß (durch Veränderung der Zahl hinter dem CLEAR-Befehl).

Jetzt geht's los

Nach dem Programmstart werden Sie zunächst aufgefordert, Ihren Namen einzugeben. Der Name darf maximal 20 Zeichen beinhalten. Sollte ein File mit dem gewählten Namen noch nicht auf Diskette vorhanden sein, legt das Programm dieses File an. Ist die Namensdatei schon auf der Diskette vorhanden, lädt der Computer alle Daten, die mit diesem Namen zusammenhängen. Hierzu gehören die Namen der Vokabeldateien auf der eingelegten Diskette, die vom Anwender eingestellten Grundfarben sowie die Benutzerdaten mit Datum und Uhrzeit. Mit Hilfe des Anwendernamens ist es möglich, eine eigene, von anderen Daten getrennte, Vokabelbibliothek aufzubauen. Diese Funktion ist interessant, wenn auf Diskette

Vokabeln eingeben Datum: 05.10.1988 Zeit: 17:37:26

Fremdwort:

Übersetzung:

franz. Zeichensatz:
F 1: è F 2: é F 3: ê
F 4: à F 5: â F 6: ù
F 7: ô F 8: ö F 9: ç
F10: î

Information:
Vokabelanzahl : 10 von maximal 200
Speicherbedarf: 5 %

Bild 2. Ein Beispiel für die Vokabelausgabe

nach einer bestimmten Vokabel gesucht wird.

Nach der Namenseingabe erscheint das Hauptmenü (Bild 1) mit Datum und Uhrzeit (rechts oben auf dem Bildschirm). Ist eine Hardware-Uhr vorhanden, übernimmt das Programm die dort abgelegten Werte. Anwender ohne Hardware-Uhr sollten das aktuelle Datum und die Uhrzeit vor Aufruf des Programms entweder über die »Preferences« oder mit Hilfe des DATE-Befehls eingeben.

Hilfreiche Maus

Das gesamte Programm ist speziell auf Maussteuerung ausgelegt, die Tastatur wird ausschließlich für Texteingaben verwendet.

Diskettenbetrieb

Der erste Punkt im Hauptmenü unterteilt sich in folgende Funktionen:

1. Vokabeldatei laden

Um eine Datei von Diskette in den Arbeitsspeicher des Computers zu laden, klicken Sie den gewünschten Dateinamen an. Sollten nicht alle gespeicherten Dateien im Ausgabefenster Platz finden, verschieben Sie mit den beiden Kästchen links unten den sichtbaren Ausschnitt.

2. Vokabeldatei speichern

Wurde eine Datei erstellt oder ergänzt, kann sie mit dieser Funktion auf Diskette gespeichert werden. Haben Sie eine neue Datei erstellt, verlangt das Programm einen Dateinamen von maximal 20 Zeichen. Soll eine vorher geladene Datei wieder abgespeichert werden, erscheint im Namensfeld die alte Bezeichnung der Datei. Mit <RETURN> übernehmen Sie diesen Namen. Wollen Sie diese Funktion abbrechen, klicken Sie das Feld »Zurück zum Hauptmenü« an. Diesen Abbruch zum Hauptmenü finden Sie bei vielen Programmfunktionen.

3. Vokabeldatei löschen

Nach Anklicken dieses Menüpunktes wird eine nicht mehr benötigte Datei von Diskette gelöscht.

4. Diskette wechseln

Sollten Sie eine neue Diskette mit Vokabeln einlegen, ist dringend erforderlich, diese Funktion anzuwählen, da das Betriebssystem ansonsten die alte Diskette zurückverlangt. Der Vokabeltrainer liest die Dateinamen erst dann von der neu eingelegten Diskette.

Vokabeln eingeben

Diese Funktion (Bild 3) erlaubt Ihnen einerseits, eine neue Vokabeldatei zu beginnen, andererseits eine schon vorhandene Datei mit neuen Vokabeln zu ergänzen. Nach Anwahl der Funktion erscheinen in der oberen Hälfte des Bildschirms zwei Eingabefelder. Hier tippen Sie das Fremdwort und die Übersetzung ein, jeweils mit anschließendem <RETURN>. Links unten sehen Sie die Belegung der Funktionstasten mit den nationalen Sonderzeichen. Voreingestellt sind hier die französischen Sonderbuchstaben. Alle vom Amiga unterstützten nationalen Zeichensätze sind auf der Workbench im Directory »DEVS/KEYMAPS« abgelegt. Wollen Sie einen anderen Zeichensatz verwenden, müssen Sie die DATA-Zeilen am Ende des Hauptprogramms verändern. Die ASCII-Codes der Sonderzeichen entnehmen Sie bitte dem Amiga-Handbuch.

Die Anzeige rechts unten zeigt Informationen über die Anzahl der Vokabeln im Speicher und die Speicherbelegung.

Verschiedene Bedeutungen einer Vokabel (Synonyme) sollten mit Kommata getrennt werden. Beispielsweise bedeuten die beiden englischen Worte »Hit« und »Beat« zu deutsch »Schlagen«. Bei der Abfrage berücksichtigt das Programm diese Synonyme. Ins Hauptmenü zurück kommen Sie mit <RETURN>, ohne eine Vokabel einzugeben.

Vokabeln ausgeben

Mit diesem Programmpunkt (Bild 2) sind Sie in der Lage, die im Speicher befindliche Datei durchzublättern, beispielsweise um sich einen Überblick über die Art der Vokabeln zu verschaffen. In den beiden Ausgabefenstern auf der lin-

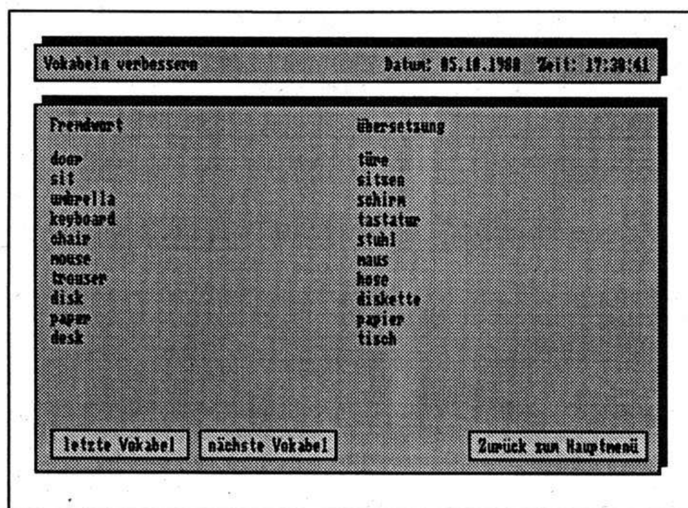


Bild 4. Hier wird die zu bearbeitende Vokabel ausgewählt

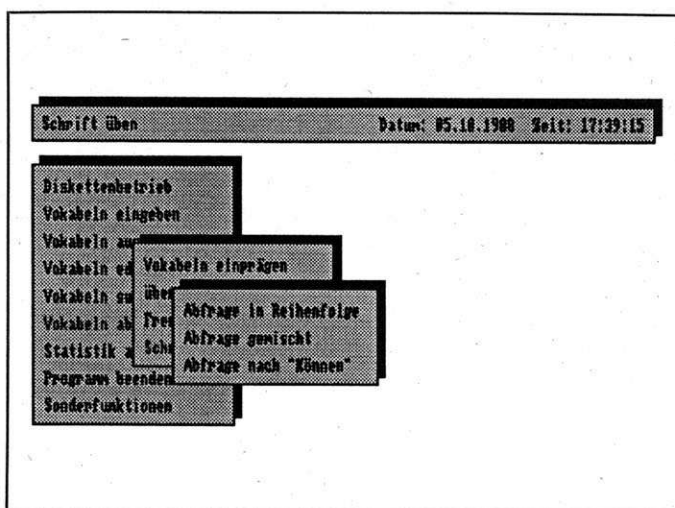


Bild 5. Diese Untermenüs erscheinen bei der Abfrage

ken Seite sehen Sie das jeweilige Fremdwort und die passende Übersetzung. Das Fenster links oben bietet Informationen über die Größe der Vokabeldatei und die laufende Nummer der angezeigten Vokabel. Der Balken unterhalb der Vokabelnummer zeigt grafisch den Anteil der Vokabeldatei in Abhängigkeit von der Maximalzahl von Vokabeln an. Im letzten Fenster rechts unten können Sie durch Anwahl von »Vokabel weiter« und »Vokabel zurück« in der Datei vor- und zurückblättern. Wenn Sie mit der Maus die Zahl »10« anklicken, wird die Datei in Zehnerschritten durchgeblättert.

Statistik ausgeben

Diese Funktion zeigt, wie häufig Sie die angezeigte Vokabel in einer bestimmten Abfrageart (s. unten) gewußt haben. Während des Durchblätterns erscheint im Balken unter dem Namen »Vokabelnummer« ein dünner roter Strich, der die aktuelle Position im Vokabelfeld markiert. Mit Hilfe der Maus läßt sich dieser Strich an eine andere Stelle des Balkens setzen. Auf diese Weise können Sie sich das Durchblättern einer längeren Datei wesentlich erleichtern.

Vokabeln editieren

Der Programmteil hilft, einzelne Vokabeln zu verbessern oder zu löschen. Die im Speicher vorhandene Datei kann unter diesem Punkt auch komplett gelöscht werden. Wollen Sie einzelne Vokabeln verbessern oder löschen, klicken Sie mit der Maus die gewünschte Vokabel an (Bild 4). Bei »Vokabel verbessern« erscheint die ausgewählte Vokabel in der gleichen Eingabemaske wie bei »Vokabel eingeben«. Der Cursor wird am Ende der Vokabel plaziert, die Verbesserungen können über Tastatur vorgenommen werden. Mit <RETURN> wird die neue Vokabel in die Datei übernommen. Durch Anwahl eines der beiden Menüpunkte »letzte Vokabel« und »nächste Vokabel« blättern Sie in der Datei.

Vokabeln suchen

Mit diesem Programmteil können Sie im Speicher oder auf Diskette eine bestimmte Vokabel finden. Soll ein Begriff auf Diskette gesucht werden, müssen Sie die zu überprüfenden Vokabeldateien auswählen. Diese Auswahl erfolgt genauso wie beim Laden oder Löschen von Dateien. Daraufhin erscheint die Aufforderung, einen Suchbegriff einzugeben. Unter Suchbegriff ist hier eine Zeichenkette zu verstehen, die an beliebiger Stelle in der gesuchten Vokabel stehen kann. Groß- oder Kleinschreibung wird bei der Suche nicht berücksichtigt. Alle Vokabeln, die den Suchbegriff beinhalten, werden ausgegeben. Im Fenster rechts oben erhalten Sie die Information, an welcher Stelle der Datei der Computer im Moment sucht. Diese Stelle wird wiederum durch einen roten Strich innerhalb der Balkenanzeige im rechten oberen Fenster markiert.

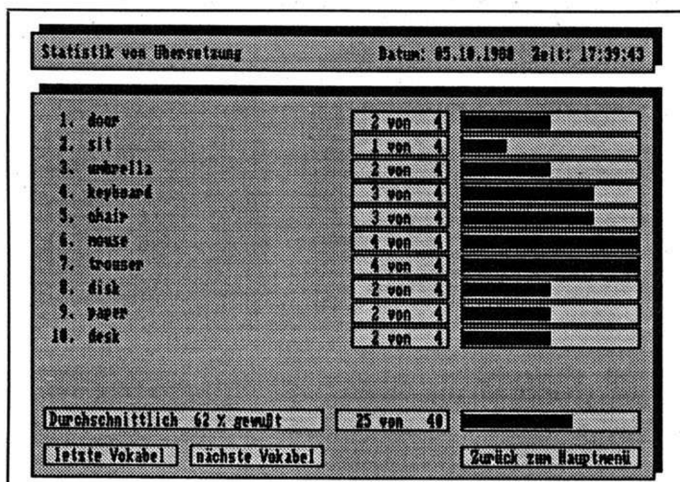


Bild 6. Statistische Auswertung einer Vokabeldatei — der Lernerfolg wird sichtbar

Frage und Antwort

Vokabeln abfragen

Dieser Menüpunkt (Bild 5) stellt Ihnen vier verschiedene Arten zur Verfügung, eine Vokabel zu lernen.

1. Vokabeln einprägen

Mit dieser Funktion können Sie sich, vor allem bei ganz neuen Vokabeln, zunächst einen Überblick verschaffen und sich die Vokabeln einprägen. Es wird noch keine statistische Speicherung vorgenommen.

2. Übersetzung abfragen
3. Fremdwort abfragen

Beide Funktionen haben den gleichen Programmablauf. Sie unterscheiden sich nur in der Art der Abfrage. Im linken oberen Fenster erscheint die jeweilige Vokabel. Nach einem Mausklick zeigt das Programm links unten das jeweilige Gegenstück der abgefragten Vokabel. Im rechten Fenster erhalten Sie die Information, die wievielte Vokabel aus der Gesamtdatei momentan abgefragt wird. Unten rechts befindet sich die Angabe über die Zahl der abgefragten Vokabeln. In diesem Fenster geben Sie auch an, ob die Vokabel bekannt war oder nicht. Sind Sie am Ende einer Datei angekommen, beginnt das Programm am Anfang der Datei. Brechen Sie die Abfrage mit dem Schalter »zurück zum hauptmenü« ab, werden die Statistikdaten der Vokabeldatei auf Diskette gespeichert.

4. Schrift üben

Bei dieser Funktion arbeitet das Programm ähnlich. Hier müssen Sie jedoch das Fremdwort über die Tastatur eingeben. Danach vergleicht der Computer die Eingabe mit dem gespeicherten Fremdwort. Er erkennt Eingaben als richtig, wenn diese entweder mit dem gespeicherten Wort übereinstimmen (Groß- und Kleinschreibung wird nicht unterschieden), oder wenn die Eingabe mit einem der Synonyme übereinstimmt.

Bei allen drei Arten der Abfrage können Sie vor Beginn die Reihenfolge festlegen, in der die Vokabeln abgearbeitet werden sollen. Die Vokabeln können in der eingegebenen Reihenfolge, in gemischter Reihenfolge oder nach Könnensstand abgefragt werden. Bei letzterem wählt der Computer primär die weniger gut beherrschten Vokabeln aus.

Statistik muß sein

Der Punkt »Statistik ausgeben« (Bild 6) erstellt für jede abgefragte Vokabel eine Statistik. Diese ist in die drei Abfragearten aufgeteilt. Die Ergebnisse sind sowohl prozentual als auch grafisch dargestellt. Mit »letzte Vokabel« und »nächste Vokabel« werden die angezeigten Worte jeweils um eine Position nach oben oder unten verschoben. Weiter zeigt die Funktion den durchschnittlichen Wert aller in einer Datei vorhandenen Vokabeln an.

Sonderfunktionen

Dieser Programmteil verleiht dem Vokabeltrainer noch mehr Komfort. Die eingestellten Farbwerte können je nach Wunsch des Anwenders verändert werden. Diese Einstellung ist nur einmal nötig, da die neu eingestellten Werte mit dem entsprechenden Anwendernamen auf Diskette gespeichert werden.

1. Info zu Abfragedaten
Zeigt an, wann die geladene Datei zum letzten Mal abgefragt wurde.

2. Benutzung des Programms
Mit diesem Punkt können Sie feststellen, an welchem Tag und zu welcher Uhrzeit Sie das Programm zuletzt benutzt haben. Diese Programmfunktion verdeutlicht, ob eine Wiederholung nach längeren Pausen nötig ist.

Programm beenden

Wichtig: Falls Sie eine Übung mit dem Vokabeltrainer beenden wollen, muß in jedem Fall diese Funktion gewählt werden. Erst nach Aktivierung dieses Punktes werden die Benutzungsdaten des Anwenders auf Diskette gespeichert. Vor dem endgültigen Ende des Programms erscheint zusätzlich eine Sicherheitsabfrage. Klicken Sie daraufhin »JA« an.

Das Programm Vokabel-Trainer benötigt außer »Trainer« und »Trainer HP« noch Amiga-Basic auf der Diskette.

Sicher wird Ihnen mit Hilfe dieses Programms das Lernen von Vokabeln mehr Freude bereiten, als nach der herkömmlichen Methode.

(Andreas Regul/ Dieter Meyer/kn/rs)

Programmname:	Trainer
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Ladeprogramm für »Trainer HP«

```
Programmname: Trainer
---------------------

EJO  REM  Ladeprogramm fuer Vokabel-Trainer
61   REM  von  Andreas Regul  (c) 7/1988
5W   REM  Initialisierung
yi4      CLEAR,100000&
FA       DIM f(16),fa(7,2)
dH       FOR i=1 TO 16
dj6        READ f(i)
DI4      NEXT
000  REM  Bildschirmaufbau
iX4      SCREEN 1,640,244,3,2
vz       WINDOW 1,"Vokabel - Trainer  Vers. 1.0
            von  Andreas Regul  (c) 7/1988",(0,0)-(631,230),0,1
Sn       FOR i=1 TO 4
nl6        MENU i,0,0,""
JO4      NEXT
by       FOR i=0 TO 7
Th6        FOR j=0 TO 2
V48          READ fa(i,j)
NS6        NEXT
2a         PALETTE i,f(fa(i,0)),f(fa(i,1)),f(fa(i,2))
PU4      NEXT
Ry       LINE (100,40)-(520,166),2,bf
2N       LINE (100,40)-(520,166),3,b
OP       LINE (101,40)-(101,166),3
OR       LINE (519,40)-(519,166),3
72       LINE (108,36)-(527,39),3,bf
Hu       LINE (521,40)-(527,162),3,bf
So       COLOR 3,2
nP       LOCATE 7,18:PRINT "V O K A B E L - T R A I N E R  Vers
         ion 1.0"
TQ       LOCATE 10,26:PRINT "v o n"
Br       LOCATE 16,44:PRINT "S o f t w a r e"
iH       LOCATE 20,18:PRINT "(w) 7/1988  Andreas Regul  Tel.: 06
         174/3706"
DU       g=36
Cs       AREA (240,112)
uS       AREA (276,76):AREA (312,112):AREA (307,112)
jE       AREA (276,81):AREA (245,112):AREA (240,112)
5B       AREAFILL
hn       AREA (255,100):AREA (298,100):AREA (299,102)
KA       AREA (256,102):AREA (256,100)
8E       AREAFILL
yL       AREA (274,90):AREA (279,90)
Yh       AREA (279,126):AREA (274,126)
BH       AREAFILL
Oq       AREA (279,90):AREA (297,90)
Z5       AREA (297,92):AREA (279,92)
EK       AREAFILL
dk       AREA (279,110):AREA (279,108)
ah       AREA (297,108):AREA (297,110)
HN       AREAFILL
55       CIRCLE (297,100),18,3,4.71,1.57
lh       CIRCLE (297,100),23,3,4.71,1.57
4z       PAINT (318,100)
Nc       LINE (298,109)-(304,109),3
66       AREA (281,108):AREA (317,126)
9w       AREA (311,126):AREA (280,111)
OU       AREAFILL
Eb       LINE (274,99)-(279,99),2
07       LINE (274,103)-(279,103),2
CE       LINE (291,90)-(293,92),2
Xd       LINE (284,90)-(286,92),2
UU       LINE (307,106)-(311,109),2
jY       LINE (301,107)-(304,110),2
sX       WHILE MOUSE(0)<0:WEND
la       t=TIMER
A5       WHILE TIMER<t+6 AND MOUSE(0)=0 AND INKEY$="":WEND
v3       RUN "TRAINER HP"
xQ0  REM  Farbwerte
TM4      DATA 0,0.04,0.1,0.16,0.22,0.29,0.35,0.41,0.47,0.54,0.6,
         0.66,0.72,0.79,0.85,0.91
xp       DATA 10,9,7,10,9,7,13,13,13,1,1,1,16,1,1,12,10,3,1,10,1
         ,1,1,15
(C) 1988 M&T
```

Listing 1. »Trainer« ist das Ladeprogramm. Es vergrößert den Basic-Speicher für das Hauptprogramm und lädt dieses nach. Bitte geben Sie das Listing mit dem Checksummer (Seite 159) ein.

Programmname:	Trainer HP
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmname: Trainer HP
------------------------

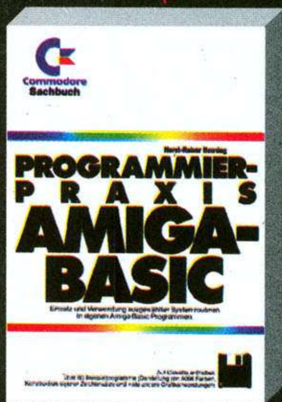
wp0   REM  Vokabel - Trainer  Vers. 1.0
N9    REM  von  Andreas Regul  (c)  7/1988
5W    REM  Initialisierung
rd4       DEFINT a-z
Du        mmv=200
Iv        DIM SHARED ta$(7),t$(100),na$(10),f!(16),farb(7,2),fa
          (10,7,2),bn$(10,2),abn$(10,2)
Is        DIM SHARED d$(101),da$(101),dm(101),ff(10),v$(mmv+1,1
          )
yu        DIM SHARED st(mmv+1,5),re(mmv+1),re1(mmv+1),vo(mmv+1)
Yw        FOR i=1 TO 7
iG6         READ ta$(i)
GL4       NEXT
dR        FOR i=1 TO 100
```

Listing 2. Das Hauptprogramm »Trainer HP« muß sich auf der Diskette im Laufwerk befinden, das als aktueller Pfad angegeben ist. Ansonsten erscheint »File not found«. Bitte mit dem Checksummer (Seite 159) eingeben.

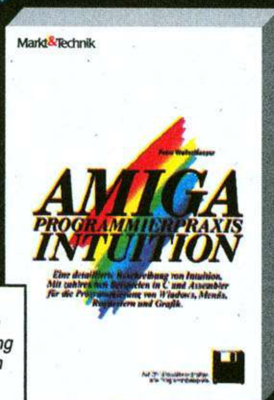
Fortsetzung des Listings auf Seite 148

Brandneue Bücher für Ihre

Amiga-Bibliothek



H. R. Henning
Programmierpraxis Amiga-Basic
Einsatz und Verwendung ausgewählter Systemroutinen in eigenen Amiga-Basic-Programmen. Die beigefügte Diskette enthält über 80 Beispielprogramme.
1988, 368 Seiten, inkl. Diskette
Bestell-Nr. 90549
ISBN 3-89090-549-8
DM 59,–/sFr 54,30/öS 460,20



P. Wollschlaeger, **Amiga: Programmierpraxis Intuition**
Eine detaillierte Beschreibung von Intuition. Mit zahlreichen Beispielen auf Diskette.
1988, 330 Seiten, inkl. Disk.
Bestell-Nr. 90593
ISBN 3-89090-593-5
DM 69,–/sFr 63,50/öS 538,20



A. Plenge
Amiga-3-D-Grafik und Animation
Eine leichtverständliche Anleitung für die Erstellung von dreidimensionalen Grafiken: Clipping, Perspektivische Projektion, Raytracing, Versteckte Linien, Schatten, Reflexion, 3-D-Editor.
1988, 376 Seiten, inkl. Diskette
Bestell-Nr. 90526, ISBN 3-89090-526-9
DM 69,–/sFr 63,50/öS 538,20



I. Krüger, **Amiga: Programmieren mit Modula 2**
Leichtverständlicher Modula-2-Kurs. Mit vielen Beispielen für die systemnahe Programmierung unter der grafischen Benutzeroberfläche »Intuition«. Auf der Diskette enthalten: alle Modula-2-Beispiele für Screens, Windows etc.
1988, 362 Seiten, inkl. Diskette
Bestell-Nr. 90554, ISBN 3-89090-554-4
DM 69,–/sFr 63,50/öS 538,20



D. Myers, **Amiga: Grafik * Musik * DFÜ**
Leichtverständlicher Programmierkurs für die erfolgreiche Grafik- und Soundprogrammierung in Amiga-Basic. Die Musikfunktionen, die Stimme des Amiga, Grafik, Animation, Datenfernübertragung und viele weitere Themen werden detailliert beschrieben.
1988, 231 Seiten, inkl. Diskette
Bestell-Nr. 90579, ISBN 3-89090-579-X
DM 59,–/sFr 54,30/öS 460,20



P. Wollschlaeger, **Amiga-Assembler-Buch**
Dieses Buch beweist, daß Assembler-Programmierung ganz einfach ist: Ein 68000er-Kurs mit vielen Beispielen. Mit ausführlichem Verzeichnis aller Systemroutinen, Anleitung für das Einbinden von Assembler-Routinen in Amiga-Basic und vielen Informationen über die Internas des Amiga-Betriebssystems. Mit Beispieldiskette.
1987, 329 Seiten, inkl. Diskette
Bestell-Nr. 90525, ISBN 3-89090-525-0
DM 59,–/sFr 54,30/öS 460,20



H. R. Henning, **Programmieren mit Amiga-Basic**
Eine gründliche Einführung in die Programmierung mit Amiga-Basic: Animation (bewegte Grafiken und Sprites) – Befehle zur Sprach- und Musikausgabe – Fenstertechnik – Sequentielle Dateiverwaltung – Spieleprogrammierung – viele Tips&Tricks und eine 3½"-Diskette mit über 100 Programmbeispielen.
1987, 363 Seiten, inkl. Diskette
Bestell-Nr. 90434, ISBN 3-89090-434-3
DM 59,–/sFr 54,30/öS 460,20

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computerfachgeschäften oder in den Fachabteilungen der Warenhäuser.

Irrtümer und Änderungen vorbehalten.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

124/807



Fragen Sie bei Ihrem Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0.
SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656,
ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0,
Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526
Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

Meister der Tasten

Sind Sie in der Lage, Ihre Amiga-Tastatur blind zu bedienen? Tippen Sie schnell und fehlerlos? Wenn ja, können Sie die nächsten Zeilen getrost überlesen. Enthaltenn Ire Texyte jedoch jede menge Tipfeler, dann ist »Keyboard-Master« genau das Richtige für Sie.

Besonders wenn Sie oft Briefe mit der Schreibmaschine oder einer Textverarbeitung verfassen oder auch hin und wieder Listings abtippen, ist es von großem Vorteil, blind schreiben zu können. Das erhöht nicht nur die Schreibgeschwindigkeit enorm, sondern hilft auch, Tippfehler zu vermeiden. Das Programm »Keyboard-Master« ermöglicht Ihnen das Erlernen des Zehn-Finger-Systems. Es stellt einen vollständigen, interaktiven Schreibmaschinen-Kurs dar, der in 20 Kapitel unterteilt ist. Die Spanne der Übungen reicht von einfachen »ASDF«-Aufgaben bis hin zu speziellen Fingerübungen für Programmierer.

Um den gewünschten Erfolg zu haben, ist es notwendig, möglichst regelmäßig zu trainieren und anfängliche Probleme durch Wiederholung der Übungen zu beseitigen. Wenn Sie das beachten, wird Ihnen dieses Lernprogramm sicherlich schnell die ersten Erfolgserlebnisse bescheren.

Die Lernerfolge mit »Keyboard-Master« sind wirklich erstaunlich. So mancher unserer Redakteure hat sich die Lektionen »reingezogen« und seine »Tipfelereien« enorm reduzieren können. Wer seine Kenntnisse im Maschineschreiben verbessern will, sollte sich das Programm unbedingt ansehen.

»Keyboard-Master« wurde in Amiga-Basic verfaßt. Trotzdem ist die Arbeitsgeschwindigkeit sehr hoch.

Das Listing

»Keyboard-Master« finden Sie auf der Programmservice-Diskette zu diesem Sonderheft. Ein Abdruck des Listings erschien uns unsinnig: Erstens ist das Programm sehr lang, und daher zum Abtippen kaum geeignet. Ein weiterer Grund sind die Übungsdateien. Wie gesagt, besteht der Kurs aus zwanzig Kapiteln mit verschiedenen Übungen. Diese müßten also, damit das Programm überhaupt lauffähig ist, ebenfalls eingegeben werden. Da dies sehr mühsam wäre, haben wir uns also entschlossen, »Keyboard-Master« im Heft nicht abzudrucken.

Was halten Sie von dieser Lösung? Sollen auch in Zukunft Listings, die zum Abdruck weniger sind, nur auf Diskette angeboten werden und damit gleichzeitig im Heft Platz für weitere Artikel/Programme geschaffen werden? Oder nehmen Sie auch das Abtippen längerer Listings in Kauf? Schreiben Sie uns Ihre Meinung! Der Fragebogen in diesem Sonderheft auf Seite 16 wartet darauf, ausgefüllt zu werden. Er bietet außerdem Gelegenheit, sich zu anderen Bereichen zu äußern. So haben Sie Gelegenheit, mit Ihren Wünschen und Anregungen die Inhalte der Sonderhefte mitzugestalten.

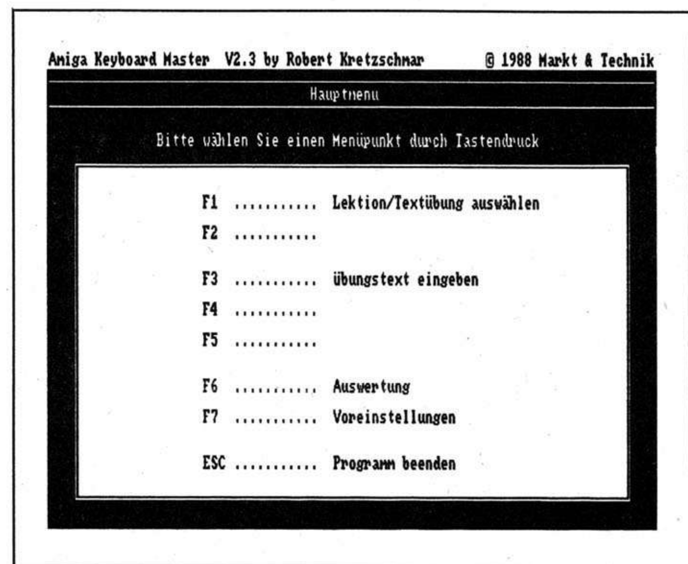


Bild 1. Das Hauptmenü von »Keyboard-Master«

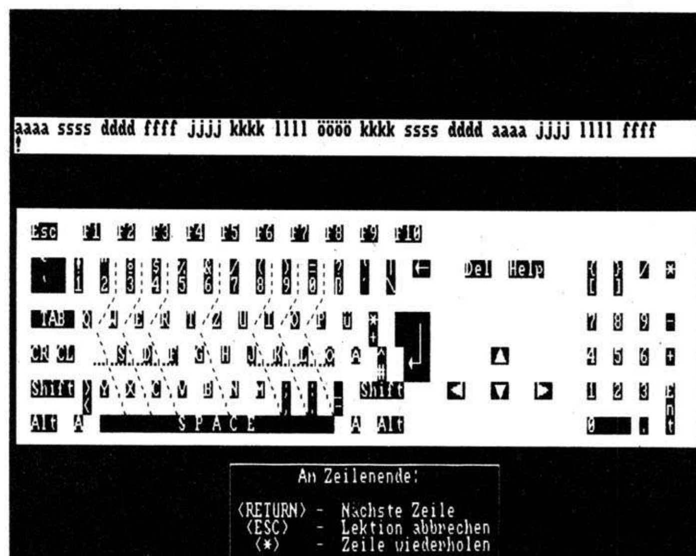


Bild 2. Die erste Übung hat begonnen

Sie finden also auf der Diskette zu diesem Sonderheft das Basic-Listing unter dem Namen »KeyboardMaster« zusammen mit den Files »Lektion 1.l« bis »Lektion 20.l« und der Übungsdatei »test.k«. Kopieren Sie alle Dateien auf eine Diskette, auf der sich Amiga-Basic befindet. Vor dem Start muß darauf geachtet werden, daß der deutsche Zeichenstatz aktiviert ist (dieser kann vom CLI aus mit »system/setmap d« aktiviert werden).

Die beiden Zeilen nach dem »CLEAR«-Befehl am Listing-Beginn enthalten zwei »Default«-Werte, die Sie Ihren Gegebenheiten anpassen können: »drive$« stellt die Laufwerksbezeichnung dar, die immer dann benutzt wird, wenn beim Laden der Kursdatei kein Laufwerk gesondert angegeben wird. Sinnvoll wäre hier, das Laufwerk (»df0:« oder »df1:«) anzugeben, von dem Sie gewöhnlich das Programm laden werden. Direkt darunter steht der voreingestellte Name der Kursdatei. In dieser Datei, die das Programm für jeden Benutzer gesondert anlegt, stehen die Informationen über die bearbeiteten Lektionen und die erzielten Ergebnisse.

Der Name dieser Datei wird beim Programmstart immer erfragt. Wenn Sie nicht jedesmal denselben Filenamen eintippen wollen, drücken Sie einfach die RETURN-Taste und der von Ihnen im Programmtext voreingestellte Name wird verwendet. Als Filename wäre zum Beispiel Ihr Vorname sinnvoll. Das Programm hängt übrigens selbständig an den eingegebenen Namen ».k« an, um es als Kursdatei von anderen Files zu unterscheiden. Beim ersten Start des Programms wird dann die Kursdatei mit dem entsprechenden Namen auf Wunsch auf Diskette erzeugt. Auf der Diskette befindet sich übrigens bereits eine Kursdatei unter dem Namen »test.k« Wenn Sie das Listing entsprechend angepaßt haben, speichern Sie es erneut. Die Arbeit mit dem Programm kann nun beginnen.

Die ersten Schritte

Nach dem Start mit »RUN« werden Sie zuerst nach dem Namen der Kursdatei gefragt (siehe oben). Wollen Sie den vorgegebenen Namen verwenden, drücken Sie nur <RETURN>. Anschließend wird die Kursdatei geladen. Wenn sie unter diesem Namen noch nicht existiert, werden Sie gefragt, ob eine neue angelegt werden soll. Beantworten Sie mit »n«, können Sie den Namen nochmals eingeben. Tippen Sie »j« (was Sie beim ersten Start tun sollten), legt das Programm eine neue Datei an. Falls Sie eine bereits existierende Kursdatei überschreiben wollen, müssen vor dem eigentlichen Filenamen drei Sterne und ein Leerzeichen eingegeben werden. Achtung: Alle Daten in dieser Datei gehen dabei verloren! Wichtig ist, daß Sie die Diskette mit der Kursdatei immer im Laufwerk lassen, solange Sie mit dem Programm arbeiten.

Sie befinden sich nun im Hauptmenü (Bild 1). Dabei wird Ihnen auffallen, daß das ganze Programm über die Tastatur gesteuert wird. Schließlich soll deren Bedienung ja erlernt werden. Zudem kann die Bedienung auch ohne Maus komfortabel und einfach sein.

Einige Menüpunkte, die im Moment noch nicht sinnvoll sind, fehlen und sind nicht wählbar. Im folgenden werden alle Menüpunkte der Reihe nach besprochen. Beginnen Sie jedoch nicht sofort, den Kurs durchzuarbeiten. Das sollten Sie erst dann tun, wenn Sie auch die Hinweise zur Durchführung des Kurses gelesen haben. Dies soll Sie aber nicht daran hindern, die Funktion des Programms beim Lesen der Bedienungsanleitung auszuprobieren.

<F1> - Lektion/Textübung wählen

Mit der Funktionstaste <F1> gelangen Sie vom Hauptmenü in ein Untermenü, in dem alle 20 Lektionen aufgelistet sind. Wählen Sie eine davon aus, indem Sie die jeweilige Zahl, gefolgt von <RETURN>, eingeben. Im Anschluß daran werden alle Lektionen in den Speicher geladen. Wenn Sie später weitere Lektionen anwählen, erfolgen keine weiteren Zugriffe auf die Diskette. Wenn Sie keine reguläre Lektion bearbeiten wollen, sondern einen beliebigen ASCII-Text üben wollen, drücken Sie statt einer Zahl nur die RETURN-Taste und Sie werden nach dem Filenamen des Textes gefragt. Tippen Sie den Namen des gewünschten Files ein. Nach dem Ladevorgang gelangen Sie wieder ins Haupt-Menü.

<F2> - Lektion starten

Je nachdem, ob Sie eine Lektion oder einen Übungstext geladen haben, hat dieser Menüpunkt den Titel »Lektion xx starten« oder »Textübung starten«. Nach Betätigen von <F2> werden zunächst einige Daten aus der Kursdatei geladen und der wichtigste Teil des Programms beginnt.

Auf dem Bildschirm wird ein Abbild einer Amiga-Tastatur, die an jene des Amiga 500 angelehnt ist, gezeichnet (Bild 2). Die Querlinien mitten im Tastenfeld sollen am Anfang helfen, alle Tasten mit dem richtigen Finger anzuschlagen. Die Hilfslinien trennen also den »Zuständigkeitsbereich« der einzelnen Finger voneinander. Die helle Linie in der Mitte stellt die Trennung zwischen rechter und linker Hand dar. Wenn in den Voreinstellungen (<F7>) das Zeichnen der Hilfslinien nicht angeschaltet ist, sind diese natürlich nicht zu sehen. Genauso verhält es sich mit der Hilfstastatur und der Anzeige der jeweils zu drückenden Taste (siehe unten).

Der Bildschirmaufbau benötigt einige Sekunden, jedoch nur beim ersten Aufruf dieses Programmteils. Danach ist die Grafik im RAM gespeichert. Ganz oben sehen Sie die Statuszeile, in der Sie lesen können, welche Lektion Sie gerade bearbeiten, welche Nummer die aktuelle Zeile hat und in welchem Teil der Lektion (Griff-, Wort- oder Satzübung) Sie sich im Moment befinden. Darunter wird die Anzahl der Fehler angezeigt, die Sie in der aktuellen Zeile gemacht haben. Noch weiter unten sehen Sie in einem farbigen Feld den Klartext der jeweiligen Zeile. Unter dieser befindet sich ein nach oben zeigender Pfeil, der auf das Zeichen zeigt, das eingetippt werden muß.

Im unteren Teil des Bildschirms befindet sich noch ein gelber Kasten, der als Hilfe für den Benutzer die möglichen Tasten am Zeilenende beinhaltet. Ist der Bilschirmaufbau abgeschlossen, ertönt ein Summen, das zeigt, daß es losgehen kann. Sie müssen nun Buchstaben für Buchstaben aus der angezeigten Textzeile abtippen. Der Pfeil wandert dabei von Zeichen zu Zeichen mit. Um Sie davon abzuhalten, spähende Blicke auf die Computertastatur zu werfen, werden die zu drückenden Tasten auf der Hilfstastatur hervorgehoben (sofern in der Voreinstellungen diese Funktion aktiviert wurde).

Machen Sie einen Fehler, hören Sie einen Piepston und der Pfeil verwandelt sich unter dem falsch getippten Zeichen in ein Ausrufungszeichen (»!«). Die Fehleranzeige wird um eins erhöht. Tippen Sie nun das Zeichen richtig, bekommt der Cursor (Pfeil) wieder sein normales Aussehen und Sie können weitertippen. Am Zeilenende angelangt erscheint an Stelle des Cursors ein Linkspfeil »←« und Sie können nun die RETURN-Taste drücken, um in die nächste Zeile zu gelangen, die ESC-Taste, um die Lektion ohne Speichern der Ergebnisse abzubrechen oder <*>, um die getippte Zeile nochmals einzugeben. In der Gesamtwertung werden dann nur jene Fehler, die beim weiteren Durcharbeiten der Zeile gemacht wurden, berücksichtigt.

Um die Funktionsweise auszutesten und sich daran zu gewöhnen, sollten Sie einmal eine Zeile abtippen, am Zeilenende aber nicht die RETURN-Taste drücken, sondern <ESC>. Schließlich handelte es sich nur um einen Versuch, und ein Speichern der Ergebnisse ist wohl kaum erwünscht.

Der Computer weiß alles...

Am Ende jeder Lektion werden die erzielten Ergebnisse, sowie allgemeine Angaben zur Lektion, am Bildschirm angezeigt (Bild 3) und in die Kursdatei gespeichert. So weiß das Programm immer, welche Lektion Sie schon durchgearbeitet haben, wieviel Fehler Sie dabei gemacht haben und wie schnell Sie beim Tippen waren. Genauso ist der Computer — und damit auch Sie — ständig im Bilde darüber, wieviel prozentual falsch gemacht wurde und wie viele Anschläge pro Minute Sie erreicht haben. Schließlich ist er auch noch darüber informiert, ob und wie oft Sie einzelne Lektionen wiederholt und sich dabei verbessert oder verschlechtert haben. Erwähnenswert ist hierbei auch die grafische Anzeige des Fehlerprozentsatzes. All das schauen Sie sich am besten erst an, wenn Sie mit dem eigentlichen Kurs ernsthaft be-


AEGIS
Sonix
AEGIS
VIDEOSCAPE
BY ALLEN HASTINGS
AEGIS:
Kreative Software
für den Amiga.

Amiga Keyboard Master V2.3 by Robert Kretzschmar © 1988 Markt & Technik

ENDE von Lektion 1

Ergebnis

Anschläge:	662
Fehler:	49
Fehlerquote:	7.40 %
benötigte Zeit:	9 Minuten, 35 Sekunden
Anschläge pro Minute:	69
Wiederholung der Lektion:	keine

Drücken Sie eine Taste

Bild 3. Die Auswertung gibt Aufschluß über den Lernerfolg und Ihren momentanen Leistungsstand

Amiga Keyboard Master V2.3 by Robert Kretzschmar © 1988 Markt & Technik

Auswertung (Kursdatei df0:test)

Lektion	Wiederh.	Fehlerquotient	Anschl./min	Differenz FQ	Punkte
1	0 mal	7.40 %	69		90
2	0 mal	5.47 %	177		112
3					
4					
5					
6	0 mal	3.81 %	150		111
7					
8					
9					
10					
11					
12					
13					
14	0 mal	6.58 %	118		97
15					
16					
17					
18					
19					
20					

Taste drücken

Bild 4. Durch das Anlegen einer Kursdatei auf Diskette wissen Sie ständig über Ihren »Wissens-Status« bescheid

ginnen wollen, denn die Bedienung dieses Programmteils erklärt sich von selbst.

<F3> - Übungstext eingeben

Dieses Lernprogramm gibt Ihnen nicht nur die Möglichkeit, fertige Lektionen zu bearbeiten, es können vielmehr auch eigene Texte eingegeben werden, die Sie zur Übung abtippen können. Dieser Programmteil ist ebenfalls weitgehend selbsterklärend. Es bleibt nur zu bemerken, daß Sie die Eingabe der Zeilen (maximal 20 Zeilen, je Zeile höchstens 78 Zeichen) mit einer Leerzeile abschließen müssen, um ins Hauptmenü zu gelangen. Dort werden Sie bemerken, daß unter »F2« nun »Textübung starten« angeführt ist. Schließlich wollen Sie jetzt Ihren Text bearbeiten, nicht die Lektion. Dasselbe gilt beim Laden eines Übungstextes (siehe oben).

<F4> - Lektions-/ Übungstext anzeigen

Über diesen Menüpunkt wird der aktuelle Lektionstext beziehungsweise der eingegebene oder geladene Übungstext angezeigt. Dabei fallen hier die violetten Zahlen am Zeilenanfang auf. Diese zeigen an, wie oft jede Zeile abgetippt werden muß.

Bei der Eingabe eines Übungstextes wird übrigens automatisch eine »1« vor jede Zeile geschrieben. Auch wenn Sie eine ASCII-Datei laden (Menüpunkt 1), wird jede Zeile, die nicht mit einer Ziffer beginnt, mit einer »1« am Zeilenanfang ergänzt. Bei den Lektionstexten können Sie mit einem ASCII-Editor die Anzahl der Wiederholungen leicht ändern. Beachten Sie jedoch, daß nur die erste Ziffer als Wiederholungszahl erkannt wird. Diesen Programmteil verlassen Sie, indem Sie eine beliebige Taste betätigen.

<F5> - Übungstext speichern

Dieser Menüpunkt kann nur dann gewählt werden, wenn Sie zuvor einen Übungstext geladen oder eingegeben haben. Das Programm fragt Sie nach dem gewünschten Filenamen. Wollen Sie den Text doch nicht speichern, drücken Sie nur <RETURN>.

<F6> - Auswertung

In diesem Programmteil werden die Daten, die die Kursdatei über die einzelnen Lektionen enthält, angezeigt (Bild 4). Lektionen, die noch nicht bearbeitet worden sind, werden dabei dunkel dargestellt. Die angewählte Lektion wird besonders gekennzeichnet. Das Feld »Differenz FQ« enthält, wenn die Lektion mehr als einmal bearbeitet worden ist, die Differenz des neueren Fehlerquotienten (FQ) zum älteren. Hier sehen Sie, ob Sie sich beim Wiederholen verbessert oder verschlechtert haben. Direkt daneben befindet sich eine weitere Spalte, die Ihnen noch unbekannt ist: Unter »Punkte« wird für jede Lektion eine bestimmte Zahl angegeben, die etwa mit einer Note vergleichbar ist. Bei Punktewerten unter 50 werden dahinter drei Ausrufungszeichen geschrieben, was bedeutet, daß Sie diese Lektion unbedingt wiederholen sollten. Werte über 100 sind sehr gut. In die Punktwertung geht neben einigen anderen Faktoren hauptsächlich der Fehlerquotient ein.

<F7> - Voreinstellungen

Mit <F7> gelangen Sie in den Programmteil, in dem Sie verschiedene Voreinstellungen (vergleichbar mit Preferences der Workbench) ändern können. Diese Änderungen werden in die Kursdatei geschrieben und bei jedem erneuten Laden der Kursdatei im Programm berücksichtigt.

Sie haben sechs Möglichkeiten, Einstellungen und damit den Programmablauf zu verändern. Es handelt sich dabei um »Schalter«, wobei durch Druck auf die links neben der jeweiligen Bezeichnung stehende Zahlentaste zwischen zwei Stellungen hin- und hergeschaltet werden kann. Bei einigen Schalterkombinationen nimmt das Programm selbständig Änderungen der Stellungen vor, das heißt es verhindert unlogische Kombinationen. Kommen wir nun zur Bedeutung der einzelnen Schalter:

Wenn Schalter 1 ausgeschaltet ist, wird die Hilfstastatur nicht mehr angezeigt.

Schalter 2 ist für die Hervorhebung der jeweils zu drückenden Tasten auf der gezeichneten Hilfstastatur zuständig.

Mit Schalter 3 wird bestimmt, ob die Textzeile immer angezeigt wird oder nicht. Es ist sinnvoll, diese auszuschalten, wenn man sich voll auf die Griffe und die Hilfstastatur konzentrieren möchte.

Ist Schalter 4 ausgeschaltet, wird das Zeichnen der Hilfslinien auf der Tastatur unterdrückt, da diese mit der Zeit störend wirken können.

Wenn Sie das automatische Wiederholen einzelner Zeilen unterdrücken wollen, müssen Sie Schalter 5 ausschalten. Dies ist dann sinnvoll, wenn Sie eine Lektion schon oft bearbeitet haben und diese nur kurz auffrischen wollen.

Mit dem sechsten und letzten Schalter wählen Sie schließlich das Default-Laufwerk (df0: oder df1:), das immer dann verwendet wird, wenn bei Filenamen keine Laufwerksangabe gesondert angegeben wird. Auch die Lektionstexte werden stets von diesem Laufwerk geladen. Bitte verwechseln Sie diese Einstellung nicht mit dem Default-Laufwerk, das Sie vor dem ersten Programmstart im Listing direkt eingestellt haben. Dieses Laufwerk gilt nur für die Kursdatei, ansonsten hat das in diesem Programmteil einzustellende Laufwerk Priorität.

Mit <ESC> werden eventuelle Änderungen in die Kursdatei gespeichert und Sie kommen zurück ins Hauptmenü.

ESC - Programm beenden

Mit <ESC> im Haupt-Menü verlassen Sie das Programm.

Der Kurs beginnt

Ich hoffe, Sie haben nun die ersten Gehversuche mit Keyboard-Master hinter sich und brennen darauf, endlich den Kurs zu beginnen.

Dazu greifen Sie zunächst, ohne das Programm zu starten, die Grundstellung, die Sie in Tabelle 1 (Seite 35) finden. Diese Fingerstellung ist sehr wichtig, prägen Sie sich diese ein und versuchen Sie mehrmals, diese zu finden, ohne auf die Tastatur zu sehen.

Sie sollten nun das Programm erneut starten und einen beliebigen Filenamen für

AMIGA MAGAZIN PROGRAMMSERVICE

Sie suchen hilfreiche Utilities und professionelle Anwendungen für Ihren Computer? Sie wünschen sich gute Software zu vernünftigen Preisen? Hier finden Sie beides!
Unser stetig wachsendes Sortiment enthält interessante Listing-Software für alle gängigen Computertypen. Jeden Monat erweitert sich unser aktuelles Angebot um eine weitere interessante Programmsammlung für jeweils einen Computertyp.
Bei Fragen zu Bestellung und Versand der Programmservice-Disketten wählen Sie bitte:
Telefon (089) 4613-232.

Bestellungen bitte *nur gegen Vorauskasse* an: Markt&Technik Verlag AG, Unternehmensbereich Buchverlag, Hans-Pinsel-Straße 2, D-8013 Haar, Telefon (089) 4613-0.
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656.
Österreich: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0, Microcomput-ique, E. Schiller, Fasangasse 24, A-1030 Wien, Telefon (0222) 785661; Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Telefon (0222) 833196. Ueberreuter Media, Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.
Bestellungen aus anderen Ländern bitte nur schriftlich an: Markt&Technik Verlag AG, Abt. Buchvertrieb, Hans-Pinsel-Straße 2, D-8013 Haar. Nur gegen Bezahlung der Rechnung im voraus.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die beigeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Amiga Sonderheft 1: Bauen Sie Ihr eigenes Tonstudio

Digisoft Plus: Die Software zu unserem Selbstbau-Digitizer, die den Amiga zum digitalen Sound-Studio macht. Das Amiga-Tonstudio arbeitet mit vier getrennten Tonspuren und besitzt neben der Digitalisierung von Klängen viele weitere Features. So können einzelne Spuren oder Teile daraus beliebig gemischt, geschnitten oder verbunden werden. Die grafische Ausgabe der digitalisierten Sounds macht die Bearbeitung zum Kinderspiel.
Suremosch: Ein Strategiespiel der neuen Dimension: Begeben Sie sich in eine ferne, fremde Welt und befreien Sie die Bevölkerung von bösen Wesen, die den Zeitfluß verlangsamen.
Dcopy: ist ein schnelles Kopierprogramm der Extraklasse. Neben hoher Geschwindigkeit bietet es die Möglichkeit, bis zu drei Kopien in einem Arbeitsgang anzufertigen.
Biorhythmus: Neben einer sehr schönen grafischen Darstellung der Kurven von Körper, Geist und Seele erlaubt das komfortable Programm auch die Ausgabe einer Jahresstatistik. Sie erfahren alles über Ihre »guten« und »kritischen« Tage.
3½"-Diskette für Amiga
Bestell-Nr. 45801 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 5/88: Vom Spiel zum nützlichen Utility

Diesmal finden Sie auf unserer Programmservice-Diskette wieder ein breites Spektrum an Listings. Von Spielen über Werkzeuge bis zu Anwendungen ist alles vorhanden.
Kniffel: Ein grafisch gut aufgemachtes Spiel für bis zu vier Teilnehmer. Kniffel wird sicher nicht langweilig. Ein Muß für alle Glücksspieler.
Manager: Verschafft Ihnen die Übersicht über Ihre Ausgaben in klarer Form und hilft somit Geld sparen. Komfortable Bedienung per Maus ist selbstverständlich.
CrossRef: Hilft Ihnen beim Analysieren von Programmen. Viele wichtige Daten von Basic-Programmen wie Labels und Variablen erhalten Sie schwarz auf weiß ausgedruckt. Ein unentbehrliches Hilfsmittel für Basic-Programmierer.
3-D-Tic-Tac-Toe: Ein gutes Auge und einen scharfen Verstand brauchen Sie für diese dreidimensionale Spielvariante.
Recover: Rettet versehentlich gelöschte Dateien von Ihrer Diskette. Auch teilweise zerstörte Dateien werden soweit als möglich restauriert.
3½"-Diskette für Amiga
Bestell-Nr. 48805 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 4/88 3-D-Landschaften aus dem Computer

Fraktalberge: Ein Muß für alle Fans von zufallserzeugten Grafiken. Fantastisch einfach in der Bedienung und sehr schnell.
Transfer: Überträgt Bilder vom C64 auf den Amiga. Mit guter Software und leicht nachzubauender Hardware.
DiskSpy: Direktes Ändern von Daten auf der Diskette ist mit diesem Werkzeug kein Problem mehr. Es stehen viele Befehle zur Verfügung.
ColorChange: Ein Basic-Unterprogramm, mit dem Sie einfach und schnell Ihre Wunschfarben auf beliebigen Bildschirmen einstellen können.
Troof: Ein spannendes Spiel in Basic mit starker Grafik und vielen verschiedenen Levels.
3½"-Diskette für Amiga
Bestell-Nr. 48804 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 3/88 Bildschirmfüllende Boot-Bilder mit allen Extras

BootGirl: Fantastische Bilder sofort nach dem Reset. Bis zu 32 Farben mit Color-Cycling. Die Bilder können auch bildschirmfüllend ohne Rand sein. Ein absolutes Muß für jeden Amiga-Besitzer.
CassCover: Selbstgedruckte Kassettenhüllen geben Ihnen den richtigen Überblick. Einfache Bedienung macht das Eingeben und Ausdrucken zur wahren Freude.
Command: Das Programm ermöglicht die Steuerung des Aztec-C-Compilers mit der Maus. Keine langen Eingaben per Tastatur, sondern ein einziger Mausklick startet nun die Übersetzung.
VideoText: Ein unentbehrliches Werkzeug für alle Video-Fans, die ihren eigenen Vorspann mit dem Amiga generieren wollen. Laufbänder, verschiedene Schriften und IFF-Bilder sind nur einige Stichpunkte, die das Programm so interessant machen.
3½"-Diskette für Amiga
Bestell-Nr. 48803 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 12/87 Super-Kopierprogramm mit viel Komfort

DCopy: Unser Programm des Monats, ein Kopierprogramm, das alles bietet, was man sich nur wünschen kann. Einige Fähigkeiten: Bis zu vier Laufwerke werden verwendet, Mehrfachkopien, abschaltbares Verify und vieles mehr.
SpeedHc: Eine sehr schnelle Hardcopyroutine für Schwarzweißausdrucke mit höchster Qualität. Leicht an andere Drucker anzupassen.
Sternenhimmel: Ein unentbehrliches Werkzeug für alle Himmelsbeobachter. Das Programm zeigt alle Sterne und Planeten von jedem beliebigen Punkt der nördlichen Hemisphäre.
Checkie42: Der Checksummer für alle Programmiersprachen von Assembler über Basic bis zu C. Ab dieser Ausgabe finden Sie bei jedem Listing die Prüfziffern.
Joy: Ein sehr kurzes und schnelles C-Programm zur Abfrage des Joysticks. Es ist leicht in eigene Programme einzubinden.
Amiga-Shell: Ein C-Programm, das Komfort ins CLI bringt. Editieren der Befehlszeile, Funktionstastenbelegung und Aliasnamen sind nur einige Fähigkeiten dieses fantastischen Programms.
3½"-Diskette für Amiga
Bestell-Nr. 48705 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 10/87 Super-Malprogramme für alle Amiga-Computer

Rainbow-Drawer: Dieses Programm des Monats bietet leistungsfähige Befehle und Funktionen, wie sie von professionellen Programmen bekannt sind: bis zu 32 Farben, alle Auflösungen, viele Befehle zum Zeichnen sowie FILL mit Mustern, BOW und anderem.
Turtle: Mit dieser Befehlserweiterung verfügen Sie über die Grafikbefehle, die bei Logo bekannt und beliebt sind.
Fractals: Dreidimensionale, realistische Gebirge mit Schattierung erzeugt dieses Programm.
Clouds: Genauso wirklichkeitsnah wie die Gebirge, aber noch erstaunlicher, sind die Wolken, die Sie mit Clouds generieren.
Apfelmännchen: Hiermit erzeugen Sie schöne Grafiken aus der beliebten Mandelbrot-Ebene.
Kudiplo: Ein gutes, unverzichtbares Werkzeug für die Kurvendiskussion stellt »Kudiplo« dar.
Senso: Testen Sie mit dieser Computer-Adaption des bekannten Spiels Ihr Gedächtnis!
Division: Bis zu 32000 Nachkommastellen können durch dieses Programm berechnet werden.
Alert: Alarme, zum Beispiel die bekannten Guru-Meditations, können Sie nun selbst programmieren. Das Programm ist in erster Linie für C-Programmierer aufschlußreich.
Border: Lassen Sie den Fensterrahmen des CLI-Fensters einfach verschwinden!
SCD: Mit diesem Utility können Sie den Pfadnamen in der Titelleiste des Fensters anzeigen.
3½"-Diskette für Amiga
Bestell-Nr. 48703 **DM 29,90*** sFr 24,90*/öS 299,–*

* Unverbindliche Preisempfehlung

Übrigens: Mit den Gutscheinen aus dem »Super-Software-Scheckheft« für DM 149,– können Sie sechs Software-Disketten Ihrer Wahl aus dem Programm-Service-Angebot der Zeitschriften

PC Magazin	Happy-Computer-Sonderheft	Computer persönlich
PC Magazin Plus	Amiga-Magazin	64'er-Magazin
Happy-Computer	Amiga-Sonderheft	64'er-Sonderheft

bestellen – egal, ob diese DM 29,90 oder DM 34,90 kosten. Das Scheckheft können Sie per Verrechnungsscheck oder mit der eingehefteten Zahlkarte direkt beim Verlag bestellen.
Kennwort: Software-Scheckheft, Bestell-Nr. 39100.

die Kursdatei (etwa Ihren Vornamen) eingeben. Wenn Sie bei den ersten Versuchen schon eine Datei unter diesem Namen erstellt haben, geben Sie drei Sterne und ein Leerzeichen vor dem eigentlichen Namen ein, so daß in jedem Fall eine völlig neue Kursdatei angelegt wird. Nun sehen Sie sich die aktuellen Voreinstellungen des Programms mit <F7> an. Ändern sollten Sie zunächst allerdings nur das Default-Laufwerk, um es Ihren Gegebenheiten anzupassen. Gehen Sie zurück ins Hauptmenü und drücken Sie <F1>. Dort geben Sie »1« gefolgt von <RETURN> ein. Nach dem Ladevorgang können Sie die Lektion starten. Bitte sehen Sie sich aber vorher die folgenden zehn Grundregeln (das sollten Sie zur besseren Einprägung möglichst vor jedem Start einer Lektion tun) an:

1. Arbeiten Sie möglichst regelmäßig mit dem Programm.
2. Bearbeiten Sie an einem Tag nur eine Lektion.
3. Wiederholen Sie jede Lektion am besten mindestens einmal.
4. Lassen Sie sich nicht von anfänglichen Schwierigkeiten entmutigen.
5. Greifen Sie die Grundstellung jedesmal blind.
6. Sehen Sie **nie** auf die Tastatur, orientieren Sie sich an der Hilfstastatur am Bildschirm.
7. Der Anschlag erfolgt kurz und ruckartig aus dem Muskel heraus.
8. Gehen Sie nach jedem Anschlag schnell in die Grundstellung zurück.
9. Tippen Sie langsam, aber gleichmäßig im Takt.
10. Halten Sie die Handgelenke möglichst hoch. Die Hand darf nicht flach auf der Tastatur liegen, sie muß angewinkelt sein.
11. Es kann nicht schaden, wenn Sie Ihre Finger mit einer speziellen Gymnastik trainieren.

Noch ein Hinweis zu Lektion 12: Beachten Sie, daß eine zu »shiftende« Taste, die sich im rechten Teil der Tastatur befindet, immer zusammen mit der linken SHIFT-Taste (und umgekehrt) angeschlagen wird. Das Programm unterstützt Sie dabei, indem immer die richtige zu drückende SHIFT-Taste hervorgehoben wird.

Haben Sie sich all dies zu Herzen genommen, kann es losgehen: Drücken Sie <F2> und tippen Sie nun alle Zeilen der Lektion ein. Schließen Sie jede Zeile mit <RETURN> ab. Werden Sie nicht bei jedem Fehler nervös. Errare humanum est! (lat.: Irren ist menschlich!) Bleiben Sie konzentriert und denken Sie daran, daß die ersten Lektionen die schwierigsten sind.

Linke Hand:	a	kleiner Finger
	s	Ringfinger
	d	Mittelfinger
	f	Zeigefinger
Rechte Hand:	j	Zeigefinger
	k	Mittelfinger
	l	Ringfinger
	ö	kleiner Finger

Tabelle 1. Die Grundstellung beim Maschineschreiben ist die Basis für alle Übungen

Sind Sie mit der Lektion fertig, werden Ihnen die Ergebnisse gezeigt. Hinter Fehler, Fehlerquotient und Anschläge pro Minute stehen je zwei Zahlen, jedoch nur dann, wenn Sie die Lektion insgesamt mindestens zweimal bearbeitet haben. Die zweite Zahl stellt jeweils die Differenz zum Ergebnis des letzten Mals dar. Die Anzeige der Anschläge je Minute sollten Sie bei den ersten Lektionen einfach übersehen. Sie sind beim Lernen völlig unwichtig und werden in der Punktewertung kaum berücksichtigt. Wichtig ist, daß Sie möglichst gleichmäßig tippen und wenige Fehler machen. Wenn Sie nun vom Hauptmenü mit <F6> die Auswertung aufrufen, können Sie sich die erreichte Punktzahl ansehen. Sie sollten diese jedoch nicht mit Schulnoten vergleichen. Denken Sie daran, daß Sie hier völlig ohne Druck lernen. Die Punkte und auch die Tippgeschwindigkeit sollen Ihnen vor allem als Vergleich dienen, wenn Sie schon einige Lektionen hinter sich haben. Punktwerte unter »50« sind jedoch ein Warnzeichen dafür, daß Sie diese Lektion wiederholen müssen. Wenn Sie mehr als »80« Punkte erreichen, können Sie sehr zufrieden sein, Werte über »100« oder gar über »120« sind sehr gut.

(Robert Kretzschmar/ M. Jobst/rs)

GRAFIK-TOOLS

Bilder lernen laufen

Filmen ist ein tolles Hobby. Wußten Sie schon, daß Sie die Ausrüstung für Computer-Movies bereits besitzen? Sie benötigen nur noch diese Listings, um den »Videorecorder« in Ihrem Amiga zu starten. Zusätzlich erhalten Sie die nötigen Basis-Informationen zur Animation von IFF-Bildern.

Vielleicht kennen Sie die Programme »Videoscape 3D« oder »Animate 3D«. Mit diesen lassen sich Animationen recht gut erstellen. Wer allerdings selbst gemalten Einzelbildern zur Bewegung verhelfen will, ist mit unserem Projekt besser bedient.

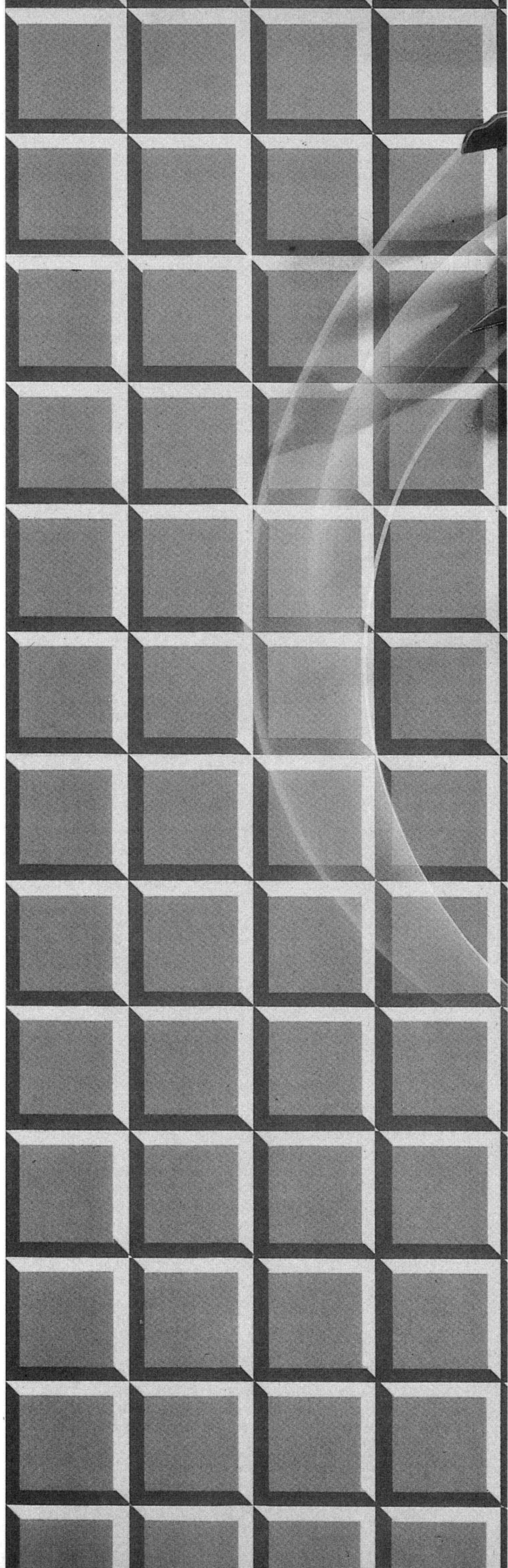
Zum Animieren Ihrer selbst erstellten Bilder benötigen Sie nur drei Dinge:
1. Ein Malprogramm, das Ihrer Kreativität freien Lauf läßt und die so entstandenen Einzelbilder der Animation im IFF-ILBM-Format abspeichern kann (z.B. »Deluxe Paint II«, »Photon Paint«, »Digi Paint«).
2. »Play« und »Anim«. Dies sind die beiden Programme, mit denen Sie fortan Ihren Amiga zum »Videorecorder« umgestalten.
3. Etwas Geduld, da das Erzeugen der Animationsdateien mitunter viel Zeit in Anspruch nehmen kann.

Bevor wir Sie nun mit der Bedienung der Programme »Play« und »Anim« (Listing 1 bis 4) vertraut machen, vorab ein wenig Theorie zum Thema Animation.

Eine Animation setzt sich aus einer bestimmten Anzahl von Einzelbildern zusammen. Die einfachste Möglichkeit diese in Bewegung zu setzen wäre, die Bilder der Reihe nach in den Bildschirmspeicher zu laden. Da hier ständig auf Diskette zugegriffen wird und zusätzlich die IFF-Bilder noch entpackt werden müßten, würde dieser Vorgang zu lange dauern — die Animation ist dann nicht mehr fließend und hätte wohl doch eher das Attribut »Diashow« verdient.

Von »Diashows« und »Page Flipping«

Weiter könnte man sämtliche Bilder erstmal in den Speicher laden und gleichzeitig entpacken, um den Diskettenzugriff zu minimieren und Rechenzeit zu sparen. Diese Methode wird auch als »Page Flipping« bezeichnet und ist natürlich äußerst schnell, solange die Bilder im Chip-Memory liegen. Wenn man allerdings bedenkt, daß eine vier Bitplanes »tiefe« und 320 x 256 Pixel große LoRes-Grafik 40 KByte Speicher benötigt, sind die Animationen bei einem Amiga mit 512 KByte Speicher leider nur sehr kurz. Es wird also sehr viel Speicher vorausgesetzt. Bei längeren Filmen müßte das Fast-Memory benutzt werden, auf das aber von den Custom-




3D

chips nicht zugegriffen werden kann. Demnach können die Bitmaps auch nicht im Fast-Memory liegen — die Bilder müßten per Prozessor zuerst »herunter kopiert« werden. »Page Flipping« sollte dann vielleicht eher als »Page Copying« bezeichnet werden. Eine längere Animation wäre aber trotz des Kopieraufwandes noch fließend. Bei einem Rechner mit großem Speicherplatz ist das »Page Flipping« daher noch zu vertreten.

Das nächste Bild bitte

Werden nun zwei aufeinanderfolgende Einzelbilder eines Films verglichen, stellt man oft fest, daß diese sich sehr ähnlich sind. Änderungen von Bild zu Bild betreffen demnach nur bestimmte Teile der Grafik. Es ist daher naheliegend, sich bei einer Animation nicht jedes Bild, sondern nur dessen Unterschied zu seinem Vorgänger zu merken.

Eine Ausnahme ist natürlich das erste Bild der Animation. Dieses hat keinen Vorgänger und muß daher vollständig vorhanden sein.

Das zuletzt beschriebene Verfahren nennt man »Delta-Animation«.

Das IFF-ANIM-File

Wir wissen jetzt, wie unser Projekt optimal arbeitet. In diesem Moment greift unser Programm »Anim« ein: Es verknüpft sozusagen die Bilderkette zu einem einzigen File. Die entstandene Datei hält sich dabei an den von Aegis definierten IFF-ANIM-Standard. Wie bei IFF-Dateien üblich, werden die Informationen in entsprechenden »Chunks« untergebracht. Die Profis kennen diesen Begriff sicher und wissen auch, was dahintersteckt. Für die weniger erfahrenen Amiga-Anwender sei hier nur erklärt, daß Chunks Standard-Informationsblöcke sind, mit denen Grafiken untereinander kompatibel bleiben. So existieren für die Differenzdaten der Bilder zwei Chunks, wobei der eine (DELTA-Chunk) die eigentlichen Daten enthält und der andere (ANHD) Aufschluß über das verwendete Kompressionsverfahren gibt. »Anim« erledigt all dies von alleine, so daß Sie sich nicht um lästige Details kümmern müssen.

Fließende Bewegungen durch »Double Buffering...«

Beim Abspielen eines Films werden normalerweise zwei Bitmaps zur Darstellung der bewegten Grafik verwendet. Die eine wird zum Anzeigen auf dem Bildschirm und die andere zum Aufbauen des neuen Bildes im Hintergrund verwendet. Man verhindert so Störungen, die beim gleichzeitigen Verändern und Anzeigen einer Grafik entstehen.

...und »interleave 2« zum schnelleren Abspielen

Nun wird es ein wenig kompliziert. Um das Abspielen einer Animationsdatei zu verstehen, sollten Sie immer folgende Grundsätze beachten:
1. Die Displaybitmap ist immer die, die gerade angezeigt wird.
2. Die Arbeitsbitmap ist diejenige, die gerade nicht angezeigt wird. Diese Bitmap kann also manipuliert werden (Double Buffering).
3. Soll das Folgebild anhand der Differenzdaten korrekt ermittelt werden, so muß in der Arbeitsbitmap genau die erste der beiden Grafiken vorliegen, die auch vorlagen, als die Differenzdaten erzeugt wurden. Nur so entspricht das Folgebild dem Bild, das beim Vergleich das zweite war.
4. Das Umschalten von einem Bild zum nächsten kann nur innerhalb der vertikalen Austastlücke geschehen.
5. Nach dem Umschalten wird die Displaybitmap zur Arbeitsbitmap und umgekehrt.

Versuchen wir einmal, den Abspielvorgang nachzuvollziehen. Es sei gerade ein Folgebild in der Arbeitsbitmap fertiggestellt. Demnach ist dieses Bild von der Reihenfolge her das Letzte. Wäre die Bilderkette durchnumeriert, so hätte es eine höhere Nummer, nennen wir sie einfach »n«. Nun wartet unser Amiga auf die Austastlücke. Nach deren Erreichen wird die Arbeitsbitmap zur Displaybitmap. Die alte Displaybitmap, deren Grafik nach Erreichen der Austastlücke mindestens einmal vom Monitor dargestellt wurde, wird nun zur neuen Arbeitsbitmap. Das Bild in unserer jetzigen Arbeitsbitmap hat demnach auch eine niedrigere Nummer, also n-1.

Wir möchten nun in dieser Arbeitsbitmap das nächste Bild erzeugen, dessen Nummer ja n+1 sein muß, da n sich gerade in der Displaybitmap befindet und demnach schon fertiggestellt ist. Gehen wir davon aus, daß bei der Bildung der Differenzdaten zwei direkt einander folgende Bilder dienten, also Bild n und Bild n+1 oder Bild n-1 und n, so muß sich nach Grundsatz 3 in unserer momentanen Arbeitsbitmap Bild n befinden, um mit den kommenden Differenzdaten Bild n+1 zu erzeugen. Es liegt aber nun Bild n-1 vor. Jetzt gibt es zwei Möglichkeiten: Entweder wird Bild n aus der Displaybitmap in die Arbeitsbitmap kopiert oder aber die Differenzdaten werden aus Bildern gebildet, die um zwei Einzelbilder auseinanderliegen (z.B. Bild n-2 und n oder Bild n-1 und n+1). Das Auseinanderliegen der Einzelbilder wird auch als Interleave bezeichnet. In den IFF-ANIM-Files verwendet man im allgemeinen einen Interleave-Faktor von 2, da somit ein Kopiervorgang wegfällt. Eine Ausnahme stellen Bild 1 und 2 dar, da Bild 1, wie schon erwähnt, als vollständige Grafik vorhanden sein muß und für das zweite noch kein um zwei Einzelbilder zurückliegendes Bild vorhanden ist.

Der CONTINOUS-Play-Mode

Bei manchen Animationen kommt es vor, daß das erste und letzte Bild identisch sind. In diesem Fall ist es nicht mehr notwendig, die Anfangsgrafik vollkommen neu aufzubauen, wie es beim Start der Animation notwendig war. Diese liegt nun in identischer Form sowieso im Speicher vor. Mit anderen Worten: Bei einem zweiten Durchlauf erhalten wir nur dann einen fließenden Übergang, wenn Start- und Endgrafik vollkommen übereinstimmen. Ist diese Bedingung nicht gegeben, so wird die Animation als »NON-CONTINOUS« bezeichnet.

Wie oben schon erläutert, wird im IFF-ANIM-Standard ein Interleave-Faktor von 2 benutzt. Konsequenterweise genügt es dann nicht, für eine CONTINOUS-Animation nur Start- und Endbild gleich zu lassen, sondern es müssen die ersten und letzten beiden Bilder identisch sein, da beim Start unseres Films ja bereits zwei Bilder als Grafikunterlage dienen (liegen n Bilder vor, so müssen 1 und n-1 sowie 2 und n übereinstimmen).

Die Anwendung: »ANIM«

Unsere beiden Programme »Play« und »Anim« können nur vom CLI aus bedient werden. Am besten kopieren Sie diese Tools in den c-Ordner oder in das aktuelle Verzeichnis. Wie schon erwähnt, dient »Anim« zur Erstellung eines IFF-Animationsfiles. Ein Aufruf könnte so aussehen:

```
anim df0:bild df1:
output.anim
```

Der Parameter »df0:bild« bezeichnet die Bilderkette. Nach Programmstart versucht »Anim«, das File »df0:bild1.pic« zu öffnen. Ist die Suche erfolgreich verlaufen, erscheint ein Screen und die Bilddaten werden eingelesen und angezeigt.

Anschließend wird nacheinander auf die Bilder »df0:bild 2.pic«, »df0:bild3.pic« bis zum letzten vorhandenen Bild der Kette zugegriffen.

Gleichzeitig werden die dabei anfallenden Differenzdaten in das File »df1:output.anim« geschrieben. Sollten die Regeln des IFF-Standards beim Einlesen eines Bildes nicht beachtet worden sein oder sind sonstige Fehler aufgetreten, wie zu wenig Speicher, wird die Routine abgebrochen.

Leider ist eine Diskette mit 880 KByte schnell voll — besonders, wenn die Animation aus Hires-Grafiken besteht. Um diesen Engpaß zu umgehen, kann »Anim« noch während des Arbeitens mit einem Malprogramm im Multitasking-Betrieb aufgerufen werden. Geben Sie einfach »run anim« vom CLI aus ein, nachdem das Malprogramm Ihrer Wahl bereits gestartet wurde. Es wird dann ein Fenster geöffnet, mit dessen Hilfe Sie sich einen Screen aus der Liste der gerade vorhandenen aussuchen können. Im Normalfall wird dies der Screen Ihres Malprogrammes sein. Anhand der gezeigten Parameter können Sie leicht sehen, welches der richtige Grafikhintergrund ist. Das »First«-Gadget zeigt den ersten, das »Next«-Gadget den nächsten Screen der Liste an. Mit »OK« bestätigt man schließlich seine Wahl

und es wird ein weiteres Fenster geöffnet. Dieses Fenster ermöglicht die Erstellung des Animationsfiles und »klaut« sich die Grafiken sozusagen aus dem gerade ausgewählten Screen.

Da dieses Fenster im Workbench-Screen geöffnet wird, muß immer zwischen diesem und Ihrem Arbeitsscreen umgeschaltet werden. Dieses erreichen Sie am einfachsten mit <Amiga-N> und <Amiga-M>.

Erstellung mit Deluxe Paint

Nehmen wir an, bei Ihrem Malprogramm handelt es sich um »Deluxe Paint II«. Entwerfen Sie nach Belieben das erste Bild des Films. Klicken Sie nun das »Save Next Frame«-Gadget an, um das erste Einzelbild zu speichern. Danach erscheint ein File-Requester, der die Auswahl des zu erzeugenden Animationsfiles ermöglicht. Ist dies geschehen, werden die Grafikdaten in dieses File geschrieben. In der Titelzeile unseres Fensters erscheint dann »Working...Please wait.«. Sie sollten dann warten, bis der Amiga seine Arbeit erledigt hat. Nach jedem weiteren erstellten Einzelbild wird wieder »Save Next Frame« betätigt, so daß jeweils immer die Differenzdaten im Output-File untergebracht werden können. Sind Sie der Meinung, das letzte Bild gemalt zu haben, wird mit »Finish« der Vorgang beendet und unser Animationsfile geschlossen, was unter keinen Umständen vergessen werden sollte! Weiter sollten Sie zwei Punkte unbedingt beachten, um sich Enttäuschungen zu ersparen:

1. Es wird genau das gespeichert, was auf dem Arbeitsscreen zu sehen ist, bei »Deluxe Paint II« also möglicherweise auch die Titelzeile und die Bedienungs-Gadgets auf der rechten Seite. Diese sind mit <F10> auszublenden (dies gilt natürlich nicht unbedingt für andere Malprogramme). Auch ein Pinsel (brush), der gerade in Gebrauch ist und mit der Maus auf der Arbeitsgrafik entlangfährt, kann möglicherweise mit im angelegten File (outputfile) landen.
2. Während in unserem Fenster die Meldung »Working...Please wait.« zu sehen ist, dürfen im Arbeitsscreen keinerlei Änderungen vorgenommen werden! Beachten Sie dies nicht, werden die Differenzdaten falsch ermittelt und die Chunks unseres Animationsfiles erhalten fehlerhafte Längen. Bei »DPaint II« müßte daher beispielsweise darauf geachtet werden, daß schon durch leichte Mausbewegungen, zum Beispiel bei einem aktivem Brush, keine Manipulation der Grafik entsteht. Dies gilt auch, wenn die Workbench im Vordergrund dargestellt wird!

Dieses Verfahren erscheint anfangs äußerst unkomfortabel. Wer aber jedes Bild speichern möchte, um es nachträglich noch verändern zu können, kommt bei langen Animationen um eine Festplatte nicht herum.

Spielereien mit »Play«

Ist der Film nun endlich in einer Datei untergebracht, können Sie mit unserem »Anim Player« das Ganze zum Leben erwecken. Nach dem oben erklärten Beispiel müßte »play df1:output.anim« eingegeben werden.

Der digitale Film wird dann in den Speicher eingelesen — sofern er sich an die IFF-Konventionen hält und genug Speicher vorhanden ist und einmal abgespielt. Wem das zuwenig ist, der kann mittels »play -L df1:output.anim« den Film in einer Endlosschleife laufen lassen. Das gilt auch bei der Eingabe von »play -C df1:output.anim«, nur sollte es sich bei »output.anim« dann um eine Continuous-Animation handeln (siehe oben).

Letztendlich kann der Trickfilm mit der linken Maustaste angehalten und mit der rechten unterbrochen werden, so daß Sie sich wieder im CLI befinden. *(Christian Wolf/rs)*

Sie benötigen zu den hier abgedruckten Listings noch das Programm »mydir.c«, das wir im Amiga-Sonderheft 1 bereits auf der Seite 63 veröffentlicht haben. Sollten Sie nicht im Besitz dieses Heftes sein, dann können Sie es bei der Redaktionsassistenz (siehe Impressum) anfordern. Auf der Programmservice-Diskette ist »mydir.c« enthalten.

Programmname:	anim.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztek-C V3.6
Aufrufe:	cc +l -s +idmp anim.c

```
Programmname: anim.c
--------------------

CVO  #define OK 1
a2   #define NEXT 3
r3   #define FIRST 4
7t   #define PLAY 5
6f   #define SAVE 6
GW   #define FINISH 7
am   #define BUFFERSIZE 4000
nr   #define NextLong ReadLongWord(MyFile)
rh   #define MAKE_ID(a,b,c,d) ( (long)(a)<<24 ] (long)(b)<
     <16 ] (c)<<8 ] (d) )
QJ   #define NOTITLE NULL
g7   #define FORM MAKE_ID('F', 'O', 'R', 'M')
6G   #define ILBM MAKE_ID('I', 'L', 'B', 'M')
pc   #define BMHD MAKE_ID('B', 'M', 'H', 'D')
fe   #define CMAP MAKE_ID('C', 'M', 'A', 'P')
AP   #define BODY MAKE_ID('B', 'O', 'D', 'Y')
Vv   #define GRAB MAKE_ID('G', 'R', 'A', 'B')
ls   #define DEST MAKE_ID('D', 'E', 'S', 'T')
df   #define SPRT MAKE_ID('S', 'P', 'R', 'T')
xc   #define CAMG MAKE_ID('C', 'A', 'M', 'G')
og   #define CRNG MAKE_ID('C', 'R', 'N', 'G')
GP   #define CCRT MAKE_ID('C', 'C', 'R', 'T')
Y6   #define DPPV MAKE_ID('D', 'P', 'P', 'V')
mL   struct BitMapHeader
8V9          UWORD w, h;
q8           UWORD x, y;
Eu           UBYTE Planes;
Oa           UBYTE masking;
iI           UBYTE compression;
Lt           UBYTE pad1;
PD           UWORD transparentColor;
uZ           UBYTE xAspect, yAspect;
Rn           WORD pageWidth, pageHeight;
```

```
qm8          };
eP0  struct BitMapHeader BitMapHeader;
8p   char ColorMap[192];
ic   char *BodyChunk;
lI   unsigned long BodySize;
HM   struct NewScreen NewS1 = {
ss7          0,0,0,0,0,0,0,0,CUSTOMSCREEN,NULL,NOTITLE,0 };
5e0  struct Screen *MyScreen;
pv   BOOL first_body=FALSE;
3t   USHORT MyLines[] = { 0,0,69,0,69,11,0,11,0,0 };
lh   USHORT PlayLines[] = { 0,0,119,0,119,11,0,11,0,0 };
OZ   USHORT SaveLines[] = { 0,0,127,0,127,11,0,11,0,0 };
xb   USHORT FinishLines[] = { 0,0,55,0,55,11,0,11,0,0 };
lL   struct Border MyBorder = { 0,0,1,3,JAM2,5,&MyLines,NULL };
Ny   struct Border PlayBorder = { 0,0,1,3,JAM2,5,&PlayLines,NUL
     L };
MU   struct Border SaveBorder = { 0,0,1,3,JAM2,5,&SaveLines,NUL
     L };
EE   struct Border FinishBorder = { 0,0,1,3,JAM2,5,&FinishLines
     ,NULL };
bK   struct IntuiText TextNext = { 1,0,JAM2,20,3,NULL,"Next",NU
     LL };
ex   struct IntuiText TextFirst = { 1,0,JAM2,16,3,NULL,"First",
     NULL };
Gw   struct IntuiText TextOk = { 1,0,JAM2,28,3,NULL,"Ok",NULL }
     ;
z2   struct IntuiText TextPlay = { 1,0,JAM2,3,2,NULL,"Play Anim
     ation",NULL };
d0   struct IntuiText TextSave = { 1,0,JAM2,3,2,NULL,"Save Next
      Frame",NULL };
zB   struct IntuiText TextFinish = { 1,0,JAM2,3,2,NULL,"Finish"
     ,NULL };
mX   struct Gadget GGNext = { NULL,85,50,70,12,GADGHCOMP,RELVER
     IFY,
```

Listing 1. »anim.c« erzeugt aus Grafiken bewegte Bilder. Bitte mit dem Checksummer (Seite 159) eingeben.

```
57 BRP                                  BOOLGADGET,&MyBorder,0,&TextNext,0,0,NEXT,0};
58 T00  struct Gadget GGFirst = { &GGNext,7,50,70,12,GADGHCOMP,RELVERIFY,
59 mKQ                                   BOOLGADGET,&MyBorder,0,&TextFirst,0,0,FIRST,0};
60 Sm0  struct Gadget GGOk = { &GGFirst,163,50,70,12,GADGHCOMP,RELVERIFY,
61 pVN                                BOOLGADGET,&MyBorder,0,&TextOk,0,0,OK,0};
62 CX0  struct Gadget GGPlay = { NULL,7,12,120,12,GADGHCOMP,RELVERIFY,
63 uNP                                  BOOLGADGET,&PlayBorder,0,&TextPlay,0,0,PLAY,0};
64 U80  struct Gadget GGSave = { &GGPlay,135,12,128,12,GADGHCOMP,RELVERIFY,
65 bnQ                                   BOOLGADGET,&SaveBorder,0,&TextSave,0,0,SAVE,0};
66 OH0  struct Gadget GGFinish = { &GGSave,271,12,56,12,GADGHCOMP,RELVERIFY,
67 pXR                                    BOOLGADGET,&FinishBorder,0,&TextFinish,0,0,FINISH,0};
68 Rd0  char MyTitle[] = "Delta Anim 0.6  1988 by CWF";
69 FZ   char Working[] = "Working...Please wait.";
70 LI   struct NewWindow NewWindow = {
71 Y97         0,0,240,66,0,1,
72 N1          GADGETUP,
73 87          ACTIVATE| WINDOWDRAG| WINDOWDEPTH,
74 4C          &GGOk,NULL,
75 fF          "Select your Screen",
76 Qu          NULL,NULL,
77 vp          640,200,640,200,
78 KO          WBENCHSCREEN };
79 p50  struct NewWindow AnimWindow = {
80 d97         0,0,334,26,0,1,
81 1Q          GADGETUP| CLOSEWINDOW,
82 r2          ACTIVATE| WINDOWDRAG| WINDOWDEPTH| WINDOWCLOSE,
83 2h          &GGFinish,NULL,
84 3L          MyTitle,
85 Z3          NULL,NULL,
86 4y          640,200,640,200,
87 TX          WBENCHSCREEN };
88 dc0  struct Window *MyWindow;
89 uN   struct IntuitionBase *IntuitionBase;
90 Er   struct GfxBase *GfxBase;
91 nZ   struct RastPort *ThisRastPort;
92 J0   char MyBuffer[40];
93 DL   char FullName[108],FileName[108];
94 Ij   APTR Buffer;
95 n9   BOOL Resident=TRUE;
96 bQ   char MyColorMap[3*64];
97 V5   FILE *MyFile;
98 vM   unsigned long offset;
99 sZ   unsigned long planes[8];
100 zU  char Pic[5]=".pic",Number[4];
101 8d  struct BitMap BackupBitMap1,BackupBitMap2;
102 SS  struct BitMap *bbm1=&BackupBitMap1,*bbm2=&BackupBitMap2;
103 nm  reverse(s)
104 m61  char s[];
105 ck   char c;
106 2w   int i,j;
107 s2   for (i=0,j=strlen(s)-1;i<j;i++,j--)
108 1J7        c=s[i];
109 Zs         s[i]=s[j];
110 HO         s[j]=c;
111 Vm0 itoa(n,s)
112 uE1  char s[];
113 nz   int n;
114 i3   int i,sign;
115 78   if ((sign = n) < 0) n=-n;
116 qB   i=0;
117 uH   do
118 tJ3    s[i++]=n%10+'0';
119 K52    } while (( n /= 10) > 0);
120 V71  if (sign < 0) s[i++]='-';
121 GY   s[i]='\0';
122 y0   reverse(s);
123 5X0 ShowScreenData(ThisScreen)
124 LT1  struct Screen *ThisScreen;
125 wV   char *ScreenName=ThisScreen->Title;
126 i1   UWORD Modes=ThisScreen->ViewPort.Modes;
127 yZ   SetDrMd(ThisRastPort,JAM2);
128 7t   SetAPen(ThisRastPort,0);
129 E2   RectFill(ThisRastPort,2,10,237,45);
130 1c   SetDrMd(ThisRastPort,JAM2);
131 D0   SetAPen(ThisRastPort,1);
132 Me   Move(ThisRastPort,7,20);
133 b3   Text(ThisRastPort,"Res: ",5);
134 ri   itoa(ThisScreen->Width,MyBuffer);
135 IU   Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
136 uj   Text(ThisRastPort," x ",3);
137 uk   itoa(ThisScreen->Height,MyBuffer);
138 LX   Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
139 qJ   Move(ThisRastPort,130,20);
140 Oa   Text(ThisRastPort,"Depth: ",7);
141 Bd   itoa(ThisScreen->RastPort.BitMap->Depth,MyBuffer);
142 Pb   Text(ThisRastPort,MyBuffer,strlen(MyBuffer));
143 f0   Move(ThisRastPort,7,32);
144 Zg   Text(ThisRastPort,"Title: ",7);
145 1E   if (ScreenName) Text(ThisRastPort,ScreenName,strlen(ScreenName));
146 hX   else Text(ThisRastPort,"untitled",8);
147 rF   Move(ThisRastPort,7,44);
148 sg   Text(ThisRastPort,"Mode: ",6);
149 4C   if (Modes & HIRES) Text(ThisRastPort,"HIRES",5);
150 XC   else if (Modes & HAM) Text(ThisRastPort,"HAM",3);
151 hW   else if (Modes & EXTRA_HALFBRITE) Text(ThisRastPort,"EXTRA_HALFBRITE",15);
152 fP   else Text(ThisRastPort,"LORES",5);
153 4P   if (Modes & LACE) Text(ThisRastPort," INTERLACE",10);
154 A40 struct Screen *GetScreen()
155 iW1  ULONG Class;
156 7A   struct IntuiMessage *MyMessage;
157 yw   struct Screen *CurrentScreen=IntuitionBase->FirstScreen;
158 yx   if (!(MyWindow=OpenWindow(&NewWindow))) return FALSE;
159 Tp   ThisRastPort=MyWindow->RPort;
160 At   ShowScreenData(CurrentScreen);
161 9q   FOREVER
162 9J3    WaitPort(MyWindow->UserPort);
163 Ik     if (MyMessage=GetMsg(MyWindow->UserPort))
164 oM8         Class=MyMessage->Class;
165 Bq          switch(Class)
166 FhA           case GADGETUP:
167 An            switch(MyMessage->IAddress->GadgetID)
168 DfC             case OK:
169 yA              CloseWindow(MyWindow);
170 uk              return(CurrentScreen);
171 aZ              case NEXT:
172 GN              if (CurrentScreen->NextScreen)
173 LFG                 ShowScreenData(CurrentScreen=CurrentScreen->NextScreen);
174 jsC             break;
175 T1              case FIRST:
176 L4              ShowScreenData(CurrentScreen=IntuitionBase->FirstScreen);
177 mv              break;
178 aU              default:
179 ox              break;
180 pyA           break;
181 dX            default:
182 r0            break;
183 PV3    ReplyMsg(MyMessage);
184 Gx0 CopyBitMap(s,d)
185 0C1  struct BitMap *s,*d;
186 tw   char p=s->Depth,j=0;
187 qW   register unsigned long c;
188 VQ   unsigned long size = s->BytesPerRow * s->Rows;
189 gl   register char *sp,*dp;
190 1T   while (p--)
191 8w9         c=size;
192 FR          sp=s->Planes[j];
193 u0          dp=d->Planes[j++];
194 i1          while (c--) *dp++=*sp++;
195 vh0 MakeColorMap(MyScreen)
196 bA1  struct Screen *MyScreen;
197 BA   UWORD *cp=MyScreen->ViewPort.ColorMap->ColorTable;
198 oN   char *cd=MyColorMap;
199 jI   UWORD Count=MyScreen->ViewPort.ColorMap->Count,c;
200 JY   for (c=0;c<Count;c++)
201 9P7       *cd++ = (*cp>>4) & 0xf0;
202 oW        *cd++ = *cp & 0xf0;
203 mw        *cd++ = *cp++ << 4;
204 Ca0 LoadColorMap(ThisView)
205 y71  struct ViewPort *ThisView;
206 3v   char *c=ColorMap;
207 4T   unsigned short i,d=1<<BitMapHeader.Planes;
```

```
VK   unsigned short r,g,b;
Eq   for (i=0;i<d;i++)
WZ7        r=*c++>>4;
B3         g=*c++>>4;
2p         b=*c++>>4;
TU         SetRGB4(ThisView,i,r,g,b);
md0  BOOL MakeBackupRaster(bm,dbm)
6G1   struct BitMap *bm,*dbm;
c7    unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->R
      ows;
Kg    unsigned short planesize=bpr*r;
yw    char j=0;
mT    APTR PlanePtr;
Kx    InitBitMap(dbm,p,bpr*8,r);
Wy    while (p--)
Q09          if (PlanePtr=AllocMem(planesize,MEMF_PUBLIC))
yME               dbm->Planes[j++]=PlanePtr;
6S9          else return(FALSE);
Mk1   return TRUE;
re0  FreeBackupRaster(bm)
HA1   struct BitMap *bm;
oJ    unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->R
      ows;
Ws    unsigned short planesize=bpr*r;
A8    char j=0;
yf    APTR PlanePtr;
Hi    while (p--) if (PlanePtr=bm->Planes[j++]) FreeMem(PlaneP
      tr,planesize);
3s0  MakeAnim(ThisScreen)
7F1   struct Screen *ThisScreen;
Oo    ULONG Class;
PS    struct IntuiMessage *MyMessage;
OW    struct BitMap *ThisBitMap=ThisScreen->RastPort.BitMap,*S
      wap;
1S    BOOL FirstTime;
jT    if (!ThisScreen) return;
n3    if (!(MyWindow=OpenWindow(&AnimWindow))) return;
Uo    if (!MakeBackupRaster(ThisBitMap,bbm1))
Xc6        FreeBackupRaster(bbm1);
AM         CloseWindow(MyWindow);
xW         return;
bw1   if (!MakeBackupRaster(ThisBitMap,bbm2))
bg6        FreeBackupRaster(bbm1);
ek         FreeBackupRaster(bbm2);
FR         CloseWindow(MyWindow);
2b         return;
aH1   FOREVER
aA3     WaitPort(MyWindow->UserPort);
jB      if (MyMessage=GetMsg(MyWindow->UserPort))
Fn8         Class=MyMessage->Class;
cH          switch(Class)
g8A           case GADGETUP:
bE            switch(MyMessage->IAddress->GadgetID)
LNC             case FINISH:
qa              if (MyFile) CloseAnim(FullName);
6F              break;
LB              case PLAY:
Un              if (MyFile) break;
G6              if (!GetFileName(FileName,FullName,"Select Pla
                y File",Buffer,BUFFERSIZE,Resident)) break;
Xr              MyFile=0;
BK              break;
JS              case SAVE:
tZ              if (!MyFile)
eiH                  if (!GetFileName(FileName,FullName,"Save
                     Animation",Buffer,BUFFERSIZE,Resident)) br
                     eak;
ch                   SetWindowTitles(MyWindow,Working,0);
r2                   if (!OpenAnim(FullName)) break;
cK                   MakeColorMap(ThisScreen);
U4                   CopyBitMap(ThisBitMap,bbm1);
X8                   CopyBitMap(ThisBitMap,bbm2);
Cr                   offset += write_ilbm(MyFile,MyColorMap,bb
                     m1,ThisScreen->ViewPort.Modes,TRUE);
gT                   SetWindowTitles(MyWindow,MyTitle,0);
yr                   FirstTime=TRUE;
5NC             else if (FirstTime)
1qH                  SetWindowTitles(MyWindow,Working,0);
nF                   offset += write_dlta(MyFile,bbm1,ThisBitM
                     ap);
cC                   CopyBitMap(ThisBitMap,bbm1);
mZ                   SetWindowTitles(MyWindow,MyTitle,0);
BR                   FirstTime=FALSE;
NAC             else
rwH                  SetWindowTitles(MyWindow,Working,0);
wP                   offset += write_dlta(MyFile,bbm2,ThisBitM
                     ap);
kL                   CopyBitMap(ThisBitMap,bbm2);
Ht                   Swap=bbm2;
S9                   bbm2=bbm1;
Ep                   bbm1=Swap;
vi                   SetWindowTitles(MyWindow,MyTitle,0);
bkC             break;
PJ              default:
dm              break;
enA           break;
AM            case CLOSEWINDOW:
RB            if (MyFile) CloseAnim(FullName);
PU            FreeBackupRaster(bbm1);
SY            FreeBackupRaster(bbm2);
3F            CloseWindow(MyWindow);
qP            return;
lu            break;
ZT            default:
nw            break;
LR3     ReplyMsg(MyMessage);
ry0  BOOL InitDisplay(bmh)
pF1   register struct BitMapHeader *bmh;
nI    if (bmh->h > 256) NewS1.ViewModes |= LACE;
xM    if (bmh->w > 352) NewS1.ViewModes |= HIRES;
C5    NewS1.ViewModes |= VP_HIDE;
RD    NewS1.Width=bmh->w;
83    NewS1.Height=bmh->h;
N5    NewS1.Depth=bmh->Planes;
HJ    if (!(MyScreen=OpenScreen(&NewS1))) return FALSE;
Pa    if (!MakeBackupRaster(MyScreen->RastPort.BitMap,bbm1)) r
      eturn FALSE;
Tf    if (!MakeBackupRaster(MyScreen->RastPort.BitMap,bbm2)) r
      eturn FALSE;
TD0  WriteBody(bmh)
Pr1   struct BitMapHeader *bmh;
1U    char *(*PlanePtr)[8]=&MyScreen->BitMap.Planes[0];
1Q    char **CurrentPtr,*ReadPointer,*pointer;
4u    unsigned short h=bmh->h,bpl=bmh->w>>3,d=bmh->Planes,
      PlaneOffset=0;
Ua    unsigned short j,count;
aV    char n,x;
0g    #asm
vkD            clr.l    d0
yo             clr.l    d1
1s             clr.l    d2
3p             move.l   _BodyChunk,a0  ;a0 = Zeiger auf Dat
               en
vu             move.l   -4(a5),a1      ;Zeiger auf Planes[0
               ]
VJ             move.w   -18(a5),d0 ; d0 = Anzahl der Zeilen
oe             move.w   -20(a5),d1 ; d1 = Bytes pro Zeile
4s             move.w   -22(a5),d2 ; d2 = Anzahl der Planes
m7             clr.l    d4         ; d4=PlaneOffset
1z             clr.l    d6         ; d6=n=0
R7             clr.l    d5         ; PlaneOffset=0
QD             clr.l    d3         ;
Kk             clr.l    d7
Tn             bra      hstart
zx0  hloop:    move.w   d2,d3
9AD            move.l   a1,a2      ;a2=CurrentPtr
Ac             bra      pstart
r70  ploop:    move.l   (a2)+,a3   ;a3=pointer
2fD            add.l    d4,a3      ; += PlaneOffset
BY             clr.b    d5         ; count=0
pW0  cloop:    clr.w    d6
GND            move.b   (a0)+,d6   ;n=ReadPointer++
qX             tst.b    d6
5E             bmi      else
kv             addq.b   #1,d6      ;n++
0x             add.b    d6,d5      ;count += n
kk             bra      cpstart
je0  cploop:   move.b   (a0)+,(a3)+
nC   cpstart:  dbra     d6,cploop
KZD            bra      cstart
WV0  else:     move.b   (a0)+,d7   ;x=ReadPointer++
3cD            neg.b    d6         ;n = -n
s3             addq.b   #1,d6      ;n++
85             add.b    d6,d5      ;count += n
```

Listing 1. »anim.c« (Fortsetzung)

```
JU                   bra     xcstart
oBO  xcloop:         move.b  d7,(a3)+
aY   xcstart:        dbra    d6,xcloop
X3   cstart:         cmp.b   d1,d5
dxD                  bne.s   cloop
iwO  pstart:         dbra    d3,ploop
MAD                  add.w   d1,d4        ;PlaneOffset+=bpl
W50  hstart:         dbra    d0,hloop
XR1    #endasm
BVO  unsigned long ReadLongWord(MyFile)
Vq1   unsigned long MyFile;
Ns    unsigned long Buffer;
fY    Read(MyFile,&Buffer,4);
Z6    return Buffer;
1ZO  BOOL ReadILBM(MyFile)
av1   unsigned long MyFile;
KZ    unsigned long ID,Length,Offset,End;
dh    BOOL Error=FALSE;
cZ    if ((ID=NextLong) == FORM)
KT6        End=NextLong-4;
3I         if ((ID=NextLong) != ILBM)
gGB             printf("Sorry, but this is not an IFF-FORM-File
                ...\n");
HW              Error=TRUE;
xk1   else
JY6        Error=TRUE;
Rs         printf("Sorry, but this is not anim IFF-ILBM-File...
           \n");
I11   Offset=0;
Oo    while (Offset < End && Error==FALSE )
n63     ID=NextLong;
cg      Length=NextLong;
t8      if (Length & 1) Length++;
iu      Offset+=Length+8;
3m      switch (ID)
nC5       case BMHD:
Gk        Read(MyFile,&BitMapHeader,sizeof(BitMapHeader));
FO        break;
Ot        case CMAP:
UK        Read(MyFile,ColorMap,Length);
IR        break;
uG        case CAMG:
4e        NewS1.ViewModes=NextLong;
LU        break;
TB        case BODY:
pv        BodySize=Length;
ZL        if (!first_body)
psA            if (!InitDisplay(&BitMapHeader))
fuF                Error=TRUE;
Ra                 break;
BKA            LoadColorMap(&MyScreen->ViewPort);
HY             first_body=TRUE;
eK5       if (!(BodyChunk=AllocMem(Length,MEMF_PUBLIC))) Error=
          TRUE;
PC        else
JPA            Read(MyFile,BodyChunk,Length);
1G             WriteBody(&BitMapHeader);
uZ             FreeMem(BodyChunk,Length);
Zi5       break;
xN        case GRAB:
rb        case DEST:
py        case SPRT:
hN        case CCRT:
8m        case CRNG:
QK        case DPPV:
Ld        Seek(MyFile,Length,0);
hq        break;
VP        default:
L4        printf("Sorry, but this is a corrupted IFF-File\n");
zE        Error=TRUE;
lu        break;
1Y1   return Error;
dlO  void MakeFileName(FileName,count)
nX1   char *FileName;
mZ    unsigned short count;
QZ    strcpy(FullName,FileName);
OD    itoa(count,Number);
qQ    strcat(FullName,Number);
m8    strcat(FullName,Pic);
GD    printf("%s\n",FullName);
kLO  BOOL Read_Files(SourceName,DestName)
Hu1   char *SourceName,*DestName;
bP    ULONG SourceFile,DestFile;
48    UWORD c=1;
pg    BOOL FirstTime=FALSE,SecondTime=FALSE;
DB    struct BitMap *Swap;
BC    while (MakeFileName(SourceName,c++),SourceFile=Open(FullN
      ame,MODE_OLDFILE))
ZV9           if (ReadILBM(SourceFile))
ceE                Close(SourceFile);
4D                 break;
8K9           if (!FirstTime)
v4E                if (!OpenAnim(DestName))
fwJ                     printf("Can't open destination file...\
                        n");
8H                      break;
kxE                MakeColorMap(MyScreen);
yR                 CopyBitMap(MyScreen->RastPort.BitMap,bbm1);
1V                 CopyBitMap(MyScreen->RastPort.BitMap,bbm2);
dk                 offset += write_ilbm(MyFile,MyColorMap,bbm1,
                   MyScreen->ViewPort.Modes,TRUE);
pi                 FirstTime=TRUE;
Ru9           else if (!SecondTime)
huE                offset += write_dlta(MyFile,bbm1,MyScreen->
                   RastPort.BitMap);
4X                 CopyBitMap(MyScreen->RastPort.BitMap,bbm1);
IT                 SecondTime=TRUE;
Cz9           else
o2E                offset += write_dlta(MyFile,bbm2,MyScreen->
                   RastPort.BitMap);
Ae                 CopyBitMap(MyScreen->RastPort.BitMap,bbm2);
5h                 Swap=bbm2;
Gx                 bbm2=bbm1;
2d                 bbm1=Swap;
xz9           Close(SourceFile);
go1   if (MyFile) CloseAnim(DestName);
8D    FreeBackupRaster(bbm1);
BH    FreeBackupRaster(bbm2);
Gy    if (MyScreen) CloseScreen(MyScreen);
cyO  main(argc,argv)
zZ1   int argc;
b9    char **argv;
wB    IntuitionBase=OpenLibrary("intuition.library",0);
su    GfxBase=OpenLibrary("graphics.library",0);
f4    argc--;
x1    argv++;
uj    if (argc==2)
VB6        Read_Files(*argv,argv[1]);
WJ1   else
cW6        if (!(Buffer=AllocMem(BUFFERSIZE,MEMF_PUBLIC)))
V9B             printf("Out of memory...\n");
BZ              exit(0);
1y6        MakeAnim(GetScreen());
zN         FreeMem(Buffer,BUFFERSIZE);
4d1   CloseLibrary(IntuitionBase);
Ek    CloseLibrary(GfxBase);
(C) 1988 M&T
```

Listing 1. »anim.c« (Schluß)

Programmname:	iff.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.6
Aufrufe:	cc +l -s +idmp iff.c

```
Programmname: iff.c
-------------------

jZO  #define MAKE_ID(a,b,c,d) ( (long)(a)<<24 ] (long)(b)<
     <16 ] (c)<<8 ] (d) )
Xy   #define FORM MAKE_ID('F', 'O', 'R', 'M')
x7   #define ILBM MAKE_ID('I', 'L', 'B', 'M')
gT   #define BMHD MAKE_ID('B', 'M', 'H', 'D')
WV   #define CMAP MAKE_ID('C', 'M', 'A', 'P')
1G   #define BODY MAKE_ID('B', 'O', 'D', 'Y')
Mm   #define GRAB MAKE_ID('G', 'R', 'A', 'B')
cj   #define DEST MAKE_ID('D', 'E', 'S', 'T')
UW   #define SPRT MAKE_ID('S', 'P', 'R', 'T')
oT   #define CAMG MAKE_ID('C', 'A', 'M', 'G')
fX   #define CRNG MAKE_ID('C', 'R', 'N', 'G')
7G   #define CCRT MAKE_ID('C', 'C', 'R', 'T')
```

```
qY   #define ANHD MAKE_ID('A', 'N', 'H', 'D')
y0   #define ANIM MAKE_ID('A', 'N', 'I', 'M')
o0   #define DLTA MAKE_ID('D', 'L', 'T', 'A')
ux   #define DIRECT 0
SS   #define XOR 1
D2   #define LONDELTA 2
N7   #define SHORTDELTA 3
mN   struct AnimHeader
I19            UBYTE operation;
uu             UBYTE mask;
hh             UWORD w,h;
0j             WORD  x,y;
hF             ULONG abstime;
Av             ULONG reltime;
8x             UBYTE interleave;
2Q             UBYTE pad[21];
mi8            };
to0  struct AnimHeader ThisAnim = { SHORTDELTA,0,0,0,0,0,1,1,1
     };
gn   union bytes4
jw8           char b4_name[4];
mX            long b4_type;
rn            };
WX0  struct iff_chunk
bh8           union bytes4 iff_type;
6Q            unsigned long iff_length;
vr            };
Za0  struct form_chunk
nn8           union bytes4 fc_type;
rw            long fc_length;
Lu            union bytes4 fc_subtype;
Ow            };
7g0  struct BitMapHeader
Tq8           UWORD w, h;
BT            UWORD x, y;
FE            UBYTE nPlanes;
Lv            UBYTE masking;
3d            UBYTE compression;
gE            UBYTE pad1;
kY            UWORD transparentColor;
Fu            UBYTE xAspect, yAspect;
m8            WORD pageWidth, pageHeight;
B7            };
Ho0  struct ILBM_info
HZ8           struct BitMapHeader header;
pi            UBYTE cmap[32*3];
cF            struct BitMap bitmap;
GC            };
pM0  extern FILE *MyFile;
hP   extern unsigned long offset,planes[8];
zx   #define DUMP    0
tL   #define RUN     1
gG   #define MINRUN 3
bV   #define MAXRUN 128
8a   #define MAXDAT 128
HK   static int pack_row(file, source, size)
EU1   FILE *file;
28    char *source;
3f    int size;
A5    char    c,
kuH                   lastc = '\0';
7W1   short mode = DUMP;
AV    short   nbuf = 0;
vo    short   rstart = 0;
jF    short putsize;
ZK    char    buf[MAXDAT*3/2];
Xu    putsize = 0;
iE    buf[0] = lastc = *source++;
LL    nbuf = 1;
sl    size--;
aM    for (; size; --size)
Le6        buf[nbuf++] = c = *source++;
11         switch (mode)
cSA             case DUMP:
SqM                         if (nbuf > MAXDAT)
RtU                                 if (file != NULL)
81c                                         if (putc(nbuf-2, fil
                                            e) == EOF)
upk                                                 return(0);
GNc                                         if (write(buf, nbuf-
                                            1, 1, file) != 1)
wrg                                             return(0);
pxU                                 putsize += nbuf;
1oQ                                 buf[0] = c;
ZZ                                  nbuf = 1;
nn                                  rstart = 0;
Tc                                  break;
sGM                         if (c == lastc)
kdQ                             if (nbuf - rstart >= MINRUN)
DAa                                     if (rstart > 0)
e6f                                         if (file != NULL)
gKk                                             if (putc(rst
                                                art-1, file)
                                                == EOF)
72p                                                    return(
                                                       0);
7mk                                             if (fwrite(b
                                                uf, rstart, 1
                                                , file) != 1)
94p                                                    return(
                                                       0);
I1j                                            putsize += rs
                                               tart+1;
j3c                                    mode = RUN;
fIQ                             else if (rstart == 0)
15U                                 mode = RUN;
aNM                         else
kkQ                             rstart = nbuf - 1;
irM                         break;
HgA             case RUN:
qSM                         if ((c != lastc) ||(nbuf - rstart >
                             MAXRUN))
sKU                                 if (file != NULL)
WMZ                                      if (putc( -(nbuf - rsta
                                         rt - 2), file) == EOF)
LGf                                              return(0);
iPZ                                      if (putc( lastc, file)
                                         == EOF)
NIf                                              return(0);
LaU                                 putsize += 2;
CFQ                             buf[0] = c;
00                              nbuf = 1;
EE                              rstart = 0;
zv                              mode = DUMP;
v4M                         break;
e66       lastc = c;
hR1   switch (mode)
I89           case DUMP:
6YH                   if (file != NULL)
hOP                           if (putc(nbuf-1, file) == EOF)
ZUX                                   return(0);
38P                           if (fwrite(buf, nbuf, 1, file) !=
                               1)
bWX                                   return(0);
X7H                   putsize += nbuf+1;
5ED               break;
e39           case RUN:
EgH                   if (file != NULL)
ncP                           if (putc( -(nbuf - rstart - 1), f
                              ile) == EOF)
hcX                                   return(0);
41P                           if (putc( lastc, file) == EOF)
jeX                                   return(0);
hwH                   putsize += 2;
DMD               break;
sJ1   return(putsize);
vv0  static unsigned long pack_bitmap(file, bm)
Tj1   FILE *file;
yr    struct BitMap *bm;
iQ0  short i, j;
sP   short row_length;
3r   unsigned short plane_offset=0;
6j   unsigned long compressed_length=0;
Lp   for (i=0; i<bm->Rows; i++)
AL8          for (j = 0; j < bm->Depth; j++)
nzG                  if ( (row_length = pack_row(file, bm->Pla
                     nes[j] + plane_offset,
3C0                          bm->BytesPerRow)) == 0)
yt                           return(0);
I7G                  compressed_length += row_length;
UG8          plane_offset += bm->BytesPerRow;
0c0  if ( compressed_length & 1)
```

Listing 2. »iff.c« enthält einige Funktionen zur Erstellung von IFF-Dateien

```
159 b38          if (file != NULL)
160 KCG                  if ( putc( 0, file) == EOF)
161 4z0                          return(0);
162 Qh8          compressed_length++;
163 u20  return(compressed_length);
164 YS   unsigned long write_ilbm(file, colors, bits, ViewMode, com
         pressed)
165 n31   FILE *file;
166 ws    unsigned char *colors;
167 q7    register struct BitMap *bits;
168 vD    unsigned long ViewMode;
169 dx    BOOL compressed;
170 aJ    struct form_chunk chunk;
171 KE    struct iff_chunk ichunk;
172 9R    struct BitMapHeader header;
173 Xn    unsigned long bits_size;
174 H4    short i,width;
175 aA    register short j;
176 Hs    register short row_offset;
177 zu    unsigned short MaxCol= 1 << bits->Depth;
178 S9    chunk.fc_type.b4_type = FORM;
179 ul    chunk.fc_subtype.b4_type = ILBM;
180 ia    if (ViewMode & HAM) MaxCol=16;
181 kX    if (ViewMode & EXTRA_HALFBRITE) MaxCol=32;
182 EV    chunk.fc_length=4+3*sizeof(struct iff_chunk)+MaxCol*3+siz
          eof(struct BitMapHeader);
183 39    if (compressed)
184 S15       if ( (bits_size = pack_bitmap(NULL, bits)) == 0) retu
              rn(0);
185 RX1   else bits_size =  bits->BytesPerRow*bits->Rows*bits->D
          epth;
186 cW    chunk.fc_length += bits_size;
187 Yd    chunk.fc_length += 12;
188 Fk0  /* if ((ViewMode & HAM) ]] (ViewMode & EXTRA_HALFBRITE)) c
         hunk.fc_length+=12; */
189 tn1   if (fwrite(&chunk, sizeof(chunk), 1, file) != 1) return(0
          );
190 B1    ichunk.iff_type.b4_type = BMHD;
191 Lb    ichunk.iff_length = sizeof(header);
192 zZ    if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
          (0);
193 Yx    header.masking = 0;
194 Iu    header.pad1 = 0;
195 Sw    header.transparentColor = 0;
196 Dj    if (compressed) header.compression = 1;
197 kg    else header.compression = 0;
198 tK    width = bits->BytesPerRow*8;
199 82    header.w = width;
200 ba    header.h = bits->Rows;
201 MS    header.nPlanes = bits->Depth;
202 bM    header.x = 0;
203 gS    header.y = 0;
204 DP    header.pageWidth = width;
205 SK    header.pageHeight = bits->Rows;
206 zl    header.xAspect = 5 * width/320;
207 90    header.yAspect = 11;
208 1v    if (fwrite(&header, sizeof(header), 1, file) != 1) return
          (0);
209 86    ichunk.iff_type.b4_type = CMAP;
210 Cz    ichunk.iff_length = MaxCol*3;
211 Is    if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
          (0);
212 kd    if (fwrite(colors, 3*MaxCol, 1, file) != 1) return(0);
213 Ju    ViewMode &= (VP_HIDE **???? 0xffff);
214 X5    ichunk.iff_type.b4_type = CAMG;
215 9u    ichunk.iff_length = 4;
216 Nx    if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
          (0);
217 P3    if (fwrite(&ViewMode, sizeof(ViewMode), 1, file) != 1) re
          turn(0);
218 sp0  /* if (ViewMode & EXTRA_HALFBRITE)
219 cA6       ichunk.iff_type.b4_type = CAMG;
220 Ez        ichunk.iff_length = 4;
221 S2        if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) r
              eturn(0);
222 U8        if (fwrite(&ViewMode, sizeof(ViewMode), 1, file) !=
              1) return(0);
223 SM5     } */
224 Z11   ichunk.iff_type.b4_type = BODY;
225 Qd    ichunk.iff_length = bits_size;
226 X7    if (fwrite(&ichunk, sizeof(ichunk), 1, file) != 1) return
          (0);
227 lr    if (compressed)
228 vT9          if (pack_bitmap(file, bits) == 0) return(0);
229 WJ1   else
230 P19          i = bits->Rows;
231 Xj           row_offset = 0;
232 3T           while (--i >= 0)
233 ZhH                  for (j=0; j<bits->Depth; j++)
234 x2P                          if (fwrite( bits->Planes[j]+row_
                                 offset, bits->BytesPerRow,
235 r3f                                     1, file) != 1)
236 HCX                                  return(0);
237 EpH                  row_offset += bits->BytesPerRow;
238 Nr1   return(chunk.fc_length+8);
239 oA0  unsigned long diff_bitmap(file,bo,bn)
240 OG1   FILE *file;
241 ev    struct BitMap *bo,*bn;
242 Pk    char p=bo->Depth,j=0;
243 zR    unsigned short planesize=(bo->BytesPerRow*bo->Rows)>>
          1;
244 Mo    unsigned short *w=0,last_offset,new_offset,start_offset,i
          ;
245 ew    register unsigned short *s,*d,plane_offset;
246 Lh    unsigned long chunk_offset=0;
247 FQ    if (file) fwrite(planes, sizeof(planes), 1, file);
248 EP    chunk_offset+=sizeof(planes);
249 yQ    while (p--)
250 FL9          planes[j]=chunk_offset;
251 nh           s=bo->Planes[j];
252 ac           d=bn->Planes[j++];
253 cd           plane_offset=0;
254 4u           last_offset=0;
255 5Q           i=0;
256 lm           if (*s != *d)
257 vKE                w=d;
258 X7                 if (file)
259 WfJ                      new_offset=0;
260 hb                       fwrite(&new_offset,2,1,file);
261 AS                       fwrite(w,2,1,file);
262 wGE                chunk_offset+=4;s++;d++;
263 Dr                 plane_offset++;
264 Y19          while (plane_offset < planesize)
265 iBH                  if (*s == *d)
266 kaM                        switch (i)
267 JZO                          case 0:
268 dA                           s++;d++;
269 GP                           break;
270 Of                           case 1:
271 kK                           if (file)
272 1QT                               new_offset=start_offset-last_
                                      offset;
273 uo                                fwrite(&new_offset,2,1,file);
274 Nf                                fwrite(w,2,1,file);
275 Ss0                          last_offset=plane_offset-1;
276 w7                           chunk_offset+=4;i=0;s++;d++;
277 OX                           break;
278 C6                           default:
279 sS                           if (file)
280 vDT                               new_offset=last_offset-start_
                                      offset-1;
281 2w                                fwrite(&new_offset,2,1,file);
282 Cu                                fwrite(&i,2,1,file);
283 EQ                                fwrite(w,2,i,file);
284 b10                          last_offset=plane_offset-1;
285 Wh                           chunk_offset += 4;chunk_offset +=
                                 (i*2);i=0;s++;d++;
286 Xg                           break;
287 SFH                  else
288 SVM                        if (!i++)
289 RqR                            w=d;
290 zD                             start_offset=plane_offset;
291 OXM                        s++;d++;
292 gKH                  plane_offset++;
293 6g9          if (file)
294 8NE                putc(0xff,file);
295 90                 putc(0xff,file);
296 S49          chunk_offset+=2;
297 Ke1   return chunk_offset;
298 E70  unsigned long write_dlta(file,bo,bn)
299 xD1   FILE *file;
300 bs    struct BitMap *bo,*bn;
301 hQ    struct form_chunk chunk;
302 RL    struct iff_chunk ichunk;
303 dt    unsigned long bits_size;
304 UB    chunk.fc_type.b4_type = FORM;
```

```
    chunk.fc_subtype.b4_type = ILBM;
    chunk.fc_length=4+2*sizeof(struct iff_chunk)+sizeof(struct AnimHeader);
    bits_size=diff_bitmap(NULL,bo,bn);
    chunk.fc_length += bits_size;
    fwrite(&chunk, sizeof(chunk), 1, file);
    ichunk.iff_type.b4_type = ANHD;
    ichunk.iff_length = sizeof(struct AnimHeader);
    fwrite(&ichunk, sizeof(ichunk), 1, file);
    fwrite(&ThisAnim, sizeof(struct AnimHeader), 1, file);
    ichunk.iff_type.b4_type = DLTA;
    ichunk.iff_length = bits_size;
    fwrite(&ichunk, sizeof(ichunk), 1, file);
    bits_size=diff_bitmap(file,bo,bn);
    return(chunk.fc_length+8);
BOOL OpenAnim(FileName)
    char *FileName;
    struct form_chunk AnimChunk;
    if (!(MyFile=fopen(FileName,"w"))) return FALSE;
    AnimChunk.fc_type.b4_type = FORM;
    AnimChunk.fc_subtype.b4_type = ANIM;
    AnimChunk.fc_length=0;
    if (fwrite(&AnimChunk, sizeof(AnimChunk), 1, MyFile) != 1) return FALSE;
    offset=4;
CloseAnim(FileName)
    char *FileName;
    ULONG Dummy;
    fclose(MyFile);
    MyFile=Open(FileName,MODE_OLDFILE);
    Read(MyFile,&Dummy,4);
    Write(MyFile,&offset,4);
    Close(MyFile);
    MyFile=0;
(C) 1988 M&T
```

Listing 2. »iff.c« (Schluß)

Programmname:	copper.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.6
Aufrufe:	cc +l -s +idmp copper.c

```
Programmname: copper.c
----------------------

/*******************************************/
/*     Copper and Display Utitlities       */
/*      written by: Christian Wolf         */
/*        Matthias-Claudius-Weg 58         */
/*           2190 Cuxhaven 1               */
/*             04721/24269                 */
/*******************************************/
#define COP1L      0x80
#define COP2L      0x84
#define COPJMP1    0x88
#define COPJMP2    0x8a
#define DIWSTRT    0x8e
#define DIWSTOP    0x90
#define DDFSTRT    0x92
#define DDFSTOP    0x94
#define BPL1P      0xe0
#define BPLCON0    0x100
#define BPLCON1    0x102
#define BPLCON2    0x104
#define BPL1MOD    0x108
#define BPL2MOD    0x10a
#define SPROP      0x120
#define COLOR00    0x180
#define INTREQ     0x9c
#define CopperEnd *((ULONG *)CurrentInstr)=0xfffffffe
UWORD *CopperList,*CurrentInstr,*LastCopperList,*DummyData,*CopperList2;
UWORD CurrentOffset=0;
char Unused[] = { "Unused" };
char *RegisterName[] =
 { "BLTDDAT","DMACONR","VPOSR","VHPOSR","DSKDATR","JOYODAT","JOY1DAT",
  "CLXDAT","ADKCONR","POTODAT","POT1DAT","POTGOR","SERDATR","DSKBYTR",
  "INTENAR","INTREQR","DSKPTH","DSKPTL","DSKPTH","DSKLEN","DSKDAT","REFPTR",
  "VPOSW","VHPOSW","COPCON","SERDAT","SERPER","POTGO","JOYTEST","STREQU",
  "STRVBL","STRHOR","STRLONG","BLTCON0","BLTCON1","BLTAFWM","BLTALWM",
  "BLTCPTH","BLTCPTL","BLTBPTH","BLTBPTL","BLTAPTH","BLTAPTL","BLTSIZE",
  Unused,Unused,Unused,"BLTCMOD","BLTBMOD","BLTAMOD","BLTDMOD",Unused,
  Unused,Unused,Unused,Unused,"BLTCDAT","BLTBDAT","BLTADAT",Unused,
  Unused,Unused,Unused,"DISKSYNC","COP1LCH","COP1LCL","COP2LCH","COP2LCL",
  "COPJMP1","COP2JMP","COPINS","DIWSTRT","DIWSTOP","DDFSTRT","DDFSTOP",
  "DMACON","CLXCON","INTENA","INTREQ","ADKCON","AUD0LCH","AUD0LCL",
  "AUD0LEN","AUD0PER","AUD0VOL","AUD0DAT",Unused,Unused,"AUD1LCH",
  "AUD1LCL","AUD1LEN","AUD1PER","AUD1VOL","AUD1DAT",Unused,Unused,
  "AUD2LCL","AUD2LCH","AUD2LEN","AUD2PER","AUD2VOL","AUD2DAT",Unused,
  Unused,"AUD3LCL","AUD3LCH","AUD3LEN","AUD3PER","AUD3VOL","AUD3DAT",
  Unused,Unused,"BPL1PTH","BPL1PTL","BPL2PTH","BPL2PTL","BPL3PTH",
  "BPL3PTL","BPL4PTH","BPL4PTL","BPL5PTH","BPL5PTL","BPL6PTH","BPL6PTL",
  Unused,Unused,Unused,Unused,"BPLCON0","BPLCON1","BPLCON2",Unused,
  "BPL1MOD","BPL2MOD",Unused,Unused,"BPL1DAT","BPL2DAT","BPL3DAT","BPL4DAT",
  "BPL5DAT","BPL6DAT",Unused,Unused,
  "SPR0PTH","SPR0PTL","SPR1PTH","SPR1PTL","SPR2PTH","SPR2PTL","SPR3PTH",
  "SPR3PTL","SPR4PTH","SPR4PTL","SPR5PTH","SPR5PTL","SPR6PTH","SPR6PTL",
  "SPR7PTH","SPR7PTL","SPR0POS","SPR0CTL","SPR0DATA","SPR0DATB","SPR1POS",
  "SPR1CTL","SPR1DATA","SPR1DATB","SPR2POS","SPR2CTL","SPR2DATA",
  "SPR2DATB","SPR3POS","SPR3CTL","SPR3DATA","SPR3DATB","SPR4POS","SPR4CTL",
  "SPR4DATA","SPR4DATB","SPR5POS","SPR5CTL","SPR5DATA","SPR5DATB","SPR6POS",
  "SPR6CTL","SPR6DATA","SPR6DATB","SPR7POS","SPR7CTL","SPR7DATA","SPR7DATB",
  "COLOR00","COLOR01","COLOR02","COLOR03","COLOR04","COLOR05","COLOR06",
  "COLOR07","COLOR08","COLOR09","COLOR10","COLOR11","COLOR12","COLOR13",
  "COLOR14","COLOR15","COLOR16","COLOR17","COLOR18","COLOR19","COLOR20",
  "COLOR21","COLOR22","COLOR23","COLOR24","COLOR25","COLOR26","COLOR27",
  "COLOR28","COLOR29","COLOR30","COLOR31" };
ShowCopperList(StartPointer)
 ULONG *StartPointer;
 ULONG Instruction;
 UWORD DataWord,Register,x,y;
 do
   Instruction = *StartPointer++;
   if (Instruction & 0x00010000)
        y = ( (Instruction >> 24) & 0xff);
        x = ( (Instruction >> 16) & 0xfe);
        if (Instruction & 0x00000001)
             printf("Skip $%x,$%x\n",x,y);
        else
             printf("WAIT $%x,$%x\n",x,y);
   else
        DataWord = Instruction & 0xffff;
```

Listing 3. »copper.c« ist ein Modul, das die Programmierung der Customchips erleichtert. Die Routine wird von »play« benötigt.

```
            Register = (Instruction >> 17) & 0xff;
            printf("MOVE $%x,%s\n",DataWord,RegisterName[Register]);
    } while ( Instruction != 0xfffffffe );
#define DiwStop(h,v) DiwStrt(h,v)
UWORD DiwStrt(h,v) UWORD h,v; { return ((h & 0xff) ] (v << 8)); }
UWORD DDFStrt(hstart,HiRes)
 UWORD hstart;
 BOOL HiRes;
 if (HiRes) return (((hstart-9)>>1) & 0xfffc);
 else return (((hstart-17)>>1) & 0xfff8);
UWORD DDFStop(ddfstrt,w) UWORD ddfstrt,w; { return (ddfstrt + (w>>1) - 8); }
CopperWait(x,y,xmask,ymask)
 UWORD x,y,xmask,ymask;
 if (y<256)
      *CurrentInstr++ = ((y << 8) ] (x >> 1)) ] 0x0001;
      *CurrentInstr++ = ((ymask << 8) ] (xmask >> 1)) & 0xfffe;
      CurrentOffset+=4;
 else
      CopperWait(448,0xff,0x1ff,0xff);
      CopperWait(x,y & 0xff,0x1ff,0xff);
CopperMove(DataWord,Register)
 UWORD DataWord,Register;
 *CurrentInstr++ = Register & 0x01fe;
 *CurrentInstr++ = DataWord;
 CurrentOffset+=4;
CopperMoveL(Address,Register)
 ULONG Address;
 UWORD Register;
 CopperMove((UWORD)(Address>>16),Register);
 CopperMove((UWORD)(Address & 0xffff),Register+2);
CopperSkip(x,y,xmask,ymask)
 UWORD x,y,xmask,ymask;
 if (y < 256)
      *CurrentInstr++ = ((y << 8) ] (x >> 1)) ] 0x0001;
      *CurrentInstr++ = ((ymask << 8) ] (xmask >> 1)) ] 0x0001;
      CurrentInstr+=4;
UWORD MakeViewMode(ViewMode,Depth)
 UWORD ViewMode,Depth;
 UWORD BplCon0=0;
 BplCon0=Depth<<12;
 if (ViewMode & HIRES) BplCon0 ]= HIRES;
 if (ViewMode & HAM) BplCon0 ]= HAM;
 if (ViewMode & DUALPF) BplCon0 ]= DUALPF;
 if (ViewMode & EXTRA_HALFBRITE) BplCon0 &= (HAM]DUALPF)**0xffff;
 if (ViewMode & LACE) BplCon0 ]= LACE;
 return BplCon0 ] (1<<9);
BOOL MakeCopperList(NewS1,NewS2,ColorMap,SecondCopperList,LOFCopperList)
 register struct NewScreen *NewS1,*NewS2;
 char *ColorMap;
 UWORD *SecondCopperList;
 UWORD *LOFCopperList;
 UWORD d,j,x,y,w=NewS1->Width,h=NewS1->Height;
 UWORD r,g,b,swap;
 struct BitMap *bm=NewS1->CustomBitMap;
 UWORD LOFModulo=0;
 UWORD OddRows=0;
 BOOL Mode=FALSE;
 unsigned char *MyPtr=ColorMap;
 if (SecondCopperList)
      LastCopperList=CurrentInstr=CopperList2=SecondCopperList;
 else if (LOFCopperList)
      CurrentInstr=LOFCopperList;
      OddRows=bm->BytesPerRow;
 else
      if (!(CopperList=AllocMem(4096,MEMF_CHIP] MEMF_CLEAR))) return FALSE;
      LastCopperList=CurrentInstr=CopperList;
 if (ColorMap)
      d=1<<NewS1->Depth;
      if (NewS1->ViewModes & HAM) d=16;
      j=0;
      while (d--)
           r=*MyPtr++>>4;
           g=*MyPtr++>>4;
           b=*MyPtr++>>4;
           CopperMove( (r << 8) ] (g << 4) ] b,COLOR00 + j);
           j+=2;
 if (h > 256) NewS1->ViewModes ]= LACE;
 if (NewS1->ViewModes & HIRES)
      Mode=TRUE;
      w >>= 1;
 if (NewS1->ViewModes & LACE)
      h >>=1;
      LOFModulo=bm->BytesPerRow;
 CopperMove(0,BPLCON1);   /*  kein Scrolling    */
 CopperMove(0,BPLCON2);   /*  Priorität egal    */
 CopperMove(MakeViewMode(NewS1->ViewModes,NewS1->Depth),BPLCON0);
 CopperMove(LOFModulo,BPL1MOD); /* Bitplanegröße==Displaygröße -> Modulo=0 */
 CopperMove(LOFModulo,BPL2MOD);
 y=41;
 x=129-(w-320);
 CopperMove(DiwStrt(x,y),DIWSTRT);
 CopperMove(DiwStop(x+w,y+h),DIWSTOP);
 CopperMove(d=DDFStrt(x,Mode),DDFSTRT);
 CopperMove(DDFStop(d,w,Mode),DDFSTOP);
 if (bm)
      d=NewS1->Depth;
      j=0;
      while (d--) CopperMoveL(bm->Planes[j++]+OddRows,BPL1P+(j<<2));
/* Alternative CopperList erstellen */
/* für interlaced Display          */
 if ((NewS1->ViewModes & LACE) && !LOFCopperList)
      CopperMoveL(CurrentInstr+6,COP1L);
      CopperEnd;
      if (!(MakeCopperList(NewS1,0,ColorMap,0,CurrentInstr+2))) return FALSE;
 else if (NewS1->ViewModes & LACE && LOFCopperList)
      CopperMoveL(LastCopperList,COP1L);
      CopperEnd;
 else CopperEnd;
/* Alternative CopperList erstellen */
/* für double buffered Display     */
 if (NewS2)
      if (!MakeCopperList(NewS2,0,ColorMap,CurrentInstr+2,0)) return FALSE;
 return TRUE;
FreeCopperList()
 if (CopperList) FreeMem(CopperList,4096);
(C) 1988 M&T
```

Listing 3. »copper.c« (Schluß)

Programmname:	play.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.6
Aufrufe:	cc +l -s +idmp play.c

```
Programmname: play.c
--------------------

#define MAKE_ID(a,b,c,d) ( (long)(a)<<24 ] (long)(b)<<16 ] (c)<<8 ] (d) )
#define NOTITLE NULL
#define FORM MAKE_ID('F', 'O', 'R', 'M')
#define ILBM MAKE_ID('I', 'L', 'B', 'M')
#define BMHD MAKE_ID('B', 'M', 'H', 'D')
#define CMAP MAKE_ID('C', 'M', 'A', 'P')
```

```
#define BODY MAKE_ID('B', 'O', 'D', 'Y')
#define GRAB MAKE_ID('G', 'R', 'A', 'B')
#define DEST MAKE_ID('D', 'E', 'S', 'T')
#define SPRT MAKE_ID('S', 'P', 'R', 'T')
#define CAMG MAKE_ID('C', 'A', 'M', 'G')
#define CRNG MAKE_ID('C', 'R', 'N', 'G')
#define CCRT MAKE_ID('C', 'C', 'R', 'T')
#define ANHD MAKE_ID('A', 'N', 'H', 'D')
#define ANIM MAKE_ID('A', 'N', 'I', 'M')
#define DLTA MAKE_ID('D', 'L', 'T', 'A')
struct FormChunk { unsigned long Form_ID;
                   unsigned long Length;
                   unsigned long ID;
                 };
struct IFFChunk { unsigned long Chunk_ID;
                  unsigned long Length;
                };
#define DIRECT 0
#define XOR 1
#define LONDELTA 2
#define SHORTDELTA 3
#define VERTDELTA 5
struct AnimHeader
          UBYTE operation;
          UBYTE mask;
          UWORD w,h;
          WORD  x,y;
          ULONG abstime;
          ULONG reltime;
          UBYTE interleave;
          UBYTE pad0;
          ULONG bits;
          UBYTE pad[18];
        };
struct BitMapHeader
          UWORD w, h;
          UWORD x, y;
          UBYTE Planes;
          UBYTE masking;
          UBYTE compression;
          UBYTE pad1;
          UWORD transparentColor;
          UBYTE xAspect, yAspect;
          WORD pageWidth, pageHeight;
        };
#define NextLong ReadLongWord(MyFile)
#define LOOP 2
#define CONT 1
#define ONCE 0
struct BitMapHeader BitMapHeader;
struct AnimHeader AnimHeader;
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct BitMap bm1,bm2;
extern UWORD *CopperList,*CopperList2;
struct NewScreen NewS1 = {
       0,0,640,256,3,0,1,HIRES,CUSTOMSCREEN,NULL,NOTITLE,0
       ,&bm1 };
struct NewScreen NewS2 = {
       0,0,640,256,3,0,1,HIRES,CUSTOMSCREEN,NULL,NOTITLE,0
       ,&bm2 };
char ColorMap[192];
struct DLTAChunk { struct DLTAChunk *Next;
                   unsigned short Size;
                 } *Start=0,*Current,*Last=0;
char *BodyChunk;
unsigned long BodySize;
BOOL Mode=ONCE,FirstTime=TRUE;
UBYTE DeltaMode;
ULONG MuluTable[626];
InitMuluTable(bpr)
 short bpr;
 short x;
 for (x=0;x<626;x++) MuluTable[x]=bpr*x;
unsigned long ReadLongWord(MyFile)
 unsigned long MyFile;
 unsigned long Buffer;
 Read(MyFile,&Buffer,4);
 return Buffer;
PlayAnim()
 MakeCopperList(&NewS1,&NewS2,ColorMap,0,0);
 Forbid();
 Disable();
 if (DeltaMode==SHORTDELTA) ShortDelta(&BitMapHeader);
 if (DeltaMode==VERTDELTA) VertDelta(&BitMapHeader);
 Permit();
 Enable();
ShortDelta(bmh)
 struct BitMapHeader *bmh;
 char *(*PlanePtr)[8]=&bm1.Planes[0];
 char **CurrentPtr=&bm2.Planes[0];
 #asm
            xdef      _CopperList,_CopperList2
FOREVER:    pea       _BitMapHeader
            jsr       _WriteBody
            add.w     #4,sp
            pea       _bm2
            pea       _bm1
            jsr       _CopyBitMap
            add.w     #8,sp
            movem.l   a4/a5,-(a7)
            move.l    _Start,a0      ;a0 = Zeiger auf Daten
            move.l    -4(a5),d7      ;Zeiger auf bm1.Planes[0]
            move.l    -8(a5),d6      ;Zeiger auf bm2.Planes[0]
            move.l    _CopperList,d5
            move.l    _CopperList2,d4
            move.w    _Mode,d2
            move.w    #%0000000110100000,$dff096 ;dma sperren
            move.l    d5,$dff080
            clr.w     $dff088
            move.w    #0111111111111111,$dff09c
            move.w    #$ffff,d3
            move.l    a0,a4
            move.w    #32,$dff09c    ; VBL-RequestBit löschen
WaitNext:   move.w    $dff01e,d0     ; auf den nächsten VBL warten
            and.w     #32,d0
            beq       WaitNext
            move.w    #$8000,$dff02a           ; LOF Bit setzen
            move.w    #%1000000110000000,$dff096 ;Copper und Plane DMA an
NextFrame:  lea       6(a0),a1       ; Zeiger auf Differenzdaten
            move.l    a1,a5          ; a1 sichern
            move.l    d6,a3          ; Zeiger auf Planes[0] in a3
            bra       dstart         ; Schleife starten
dloop:      move.l    d0,a2          ; Offset in a2
            adda.l    a5,a2          ; absolute Adresse berechnen
            move.l    (a3)+,a6       ; a6=Zeiger auf Bitplane
wloop:      move.w    (a2)+,d1       ; Datenwort holen
            bmi       next           ; negativ ? ja -> next
            add.w     d1,d1          ; d1 *= 2
            add.w     d1,a6          ; Adresse des Wortes zu a6 addieren
            move.w    (a2)+,(a6)     ; in Bitplane schreiben
            bra       wloop          ; nächstes Wort
next:       eor.w     d3,d1
            beq       dstart         ;d1=0 ? ja -> next plane
            add.w     d1,d1          ; d1 verdoppeln
            add.w     d1,a6          ; Adresse des ersten Wortes in der Bpl
            move.w    (a2)+,d0       ; Anzahl holen und ...
            bra       copystart
copyloop:   move.w    (a2)+,(a6)+
copystart:  dbra      d0,copyloop    ; kopieren
```

Listing 4. Mit »play.c« lernen die von »anim.c« erzeugten Bilder laufen

```
lnD                 suba.w    #2,a6          ; BitplaneZeiger korrigieren
zD                  bra       wloop          ; weiter geht's
eI0  dstart:        move.l    (a1)+,d0       ; Zeiger auf Daten innerhalb des Chunks
EKD                 bne.s     dloop          ; ungleich 0 ? ja -> nächste Plane
Rw                  and.w     #1024,$dff016 ; rechte Maustaste gedrückt ?
35                  beq       Exit
bP                  andi.b    #64,$bfe001 ; linke Maustaste gedrückt
Hj                  bne       skip
Oc0  WaitMouse1:    andi.b    #64,$bfe001
ADD                 beq       WaitMouse1
JE0  skip:          exg       d6,d7          ; Zeiger auf Planes[0] austauschen
RyD                 exg       d4,d5          ; Zeiger auf CopperList austauschen
x7                  move.l    (a0),a0        ; Ende der Animation
pR                  cmpa.l    #0,a0          ; erreicht ???
Xs                  bne.s     NotEnd
YH                  cmpi.w    #0,d2
OI                  beq.s     Exit
dN                  cmpi.w    #1,d2
HB                  beq.s     Loop
Kl                  movem.l   (a7)+,a4/a5
44                  move.w    #%0000000110100000,$dff096 ;DMA sperren
We                  bra       FOREVER
VC0  Loop:          move.l    a4,a0          ; wieder von Anfang an
8aD                 move.l    (a0),a0
8g0  NotEnd:        move.w    #32,$dff09c
pV   WaitFrame:     move.w    $dff01e,d0      ; auf nächsten
BjD                 and.w     #32,d0          ; VBL
Qt                  beq       WaitFrame      ; warten...
3x                  move.w    #$8000,$dff02a  ; LOF Bit setzen (falls Interlace)
1t                  move.l    d5,$dff080     ; neue Copperliste
QN                  clr.w     $dff088        ; und in CopperPC laden
vn                  bra.s     NextFrame      ; nächstes Bild bearbeiten
4j0  Exit:          movem.l   (a7)+,a4/a5
HHD                 move.w    #%0000000110100000,$dff096 ;DMA sperren
Gv                  move.l    _GfxBase,a0    ; alte CopperList
sd                  move.l    38(a0),$dff080  ; zurück schreiben
RG                  clr.w     $dff088        ; und starten
JV                  move.w    #%1000000110100000,$dff096 ; DMA wieder an
bV1   #endasm
ALO  VertDelta(bmh)
Ik1   struct BitMapHeader *bmh;
O1    char *(*PlanePtr)[8]=&bm1.Planes[0];
bN    char **CurrentPtr=&bm2.Planes[0];
FR    short BytesPerRow=((bmh->w)>>4)<<1;
Kp    ULONG Swap1,Swap2,Start;
sY    #asm
7CD                 xdef      _CopperList,_CopperList2
Du                  move.l    _CopperList,-14(a5)
Hq                  move.l    _CopperList2,-18(a5)
PQ0  FOREVER1:      pea       _BitMapHeader
u2D                 jsr       _WriteBody
1M                  add.w     #4,sp
w0                  pea       _bm2
sv                  pea       _bm1
LV                  jsr       _CopyBitMap
1g                  add.w     #8,sp
qt                  movem.l   a4/a5,-(a7)
sz                  move.l    _Start,a0      ;a0 = Zeiger auf Daten
Av                  move.w    _Mode,d2
PN                  move.w    #%0000000110100000,$dff096 ;dma sperren
Bh                  move.l    _CopperList,$dff080
4T                  clr.w     $dff088
fm                  move.w    #0111111111111111,$dff09c
kr                  move.l    a0,-22(a5)
4Q                  move.w    -10(a5),d4     ; d4=Anzahl der Spalten
8f                  lea       _MuluTable,a4
Be                  move.w    #32,$dff09c    ; VBL-RequestBit löschen
aW0  WaitNext1:     and.w     #32,$dff01e    ; auf den nächsten VBL warten
GPD                 beq       WaitNext1
x2                  move.w    #$8000,$dff02a              ; LOF Bit setzen
iy                  move.w    #%1000000110000000,$dff096 ;Copper und Plane DMA an
iv0  NextFrame1:    lea       6(a0),a1       ; Zeiger auf Differenzdaten
5PD                 move.l    a1,d5          ; a1 sichern
Hy                  move.l    -8(a5),a3      ; Zeiger auf Planes[0] in a3
Fp                  bra       dstart1        ; Schleife starten
au0  dloop1:        move.l    d0,a2          ; Offset in a2
BmD                 adda.l    d5,a2          ; absolute Adresse berechnen
tv                  move.l    (a3)+,d6       ; d6=Zeiger auf Bitplane
VT                  move.w    d4,d3          ; d3=Zähler für Spalten
NL                  subq.w    #1,d3
1K0  sloop:         clr.w     d0
AOD                 move.b    (a2)+,d0       ; Anzahl der Opcodes
2x                  move.l    d6,a6          ; Zeiger auf Bitplane in a6
Vc                  bra       OpStart        ; Schleife starten
jh0  OpLoop:        clr.w     d1             ; d1 löschen
eSD                 move.b    (a2)+,d1       ; Opcode holen
It                  bne.s     not_same       ; ungleich 0 ? ja -> not_same
ox                  move.b    (a2)+,d1       ; Opcode = same : d1=Anzahl
KQ                  subq.w    #1,d1
bF                  move.b    (a2)+,d7       ; Datum holen
870  floop:         move.b    d7,(a6)        ; und kopieren
5SD                 adda.w    d4,a6          ; Zeiger um Spaltenanzahl erhöhen
Wy                  dbra      d1,floop       ; nächstes Bytes
p3                  dbra      d0,OpLoop
TD                  addq      #1,d6
Gf                  dbra      d3,sloop
Jp                  move.l    (a1)+,d0       ; Zeiger auf Daten innerhalb des Chunks
N3                  bne.s     dloop1         ; ungleich 0 ? ja -> nächste Plane
Da                  bra       RMouse
vj0  not_same:      bmi.s     uniq           ; bit 7 gesetzt ? ja -> uniq
CcD                 add.w     d1,d1
Pb                  add.w     d1,d1          ; d1 *= 4
BL                  adda.l    (a4,d1),a6
zD                  dbra      d0,OpLoop
dN                  addq      #1,d6
Qp                  dbra      d3,sloop
Dr                  move.l    (a1)+,d0       ; Zeiger auf Daten innerhalb des Chunks
v4                  bne.s     dloop1         ; ungleich 0 ? ja -> nächste Plane
Nk                  bra       RMouse
nX0  uniq:          and.b     #127,d1        ; opcode = uniq : Bit 7 löschen
gmD                 subq.w    #1,d1
Zt0  copyloop1:     move.b    (a2)+,(a6)
mVD                 adda.w    d4,a6          ; a6 um Spaltenanzahl erhöhen
8u                  dbra      d1,copyloop1
RG0  OpStart:       dbra      d0,OpLoop
oYD                 addq      #1,d6
ME0  sstart:        dbra      d3,sloop
NN   dstart1:       move.l    (a1)+,d0       ; Zeiger auf Daten innerhalb des Chunks
10D                 bne.s     dloop1         ; ungleich 0 ? ja -> nächste Plane
k00  RMouse:        and.w     #1024,$dff016 ; rechte Maustaste gedrückt ?
YUD                 beq       Exit1
XY                  andi.b    #64,$bfe001    ; linke Maustaste gedrückt
m8                  bne       skip1
LV0  WaitMouse2:    andi.b    #64,$bfe001
```

```
                beq      WaitMouse2
skip1:          move.l   -4(a5),d7
                move.l   -8(a5),-4(a5)
                move.l   d7,-8(a5)       ; Zeiger auf Planes [0] austauschen
                move.l   -14(a5),d7
                move.l   -18(a5),-14(a5)
                move.l   d7,-18(a5)  ; Zeiger auf CopperList austauschen
                move.l   (a0),a0     ; Ende der Animation
                cmpa.l   #0,a0       ; erreicht ???
                bne.s    NotEnd1
                cmpi.w   #0,d2
                beq.s    Exit1
                cmpi.w   #1,d2
                beq.s    Loop1
                movem.l  (a7)+,a4/a5
                move.w   #%0000000110100000,$dff096 ;DMA sperren
                bra      FOREVER1
Loop1:          move.l   -22(a5),a0      ; wieder von Anfang an
                move.l   (a0),a0
NotEnd1:        move.w   #32,$dff09c
WaitFrame1:     and.w    #32,$dff01e     ; auf nächsten
                beq      WaitFrame1      ; warten...
                move.w   #$8000,$dff02a  ; LOF Bit setzen (falls Interlace)
                move.l   -14(a5),$dff080     ; neue Copperliste setzen
                clr.w    $dff088         ; und in CopperPC laden
                bra.s    NextFrame1      ; nächstes Bild bearbeiten
Exit1:          movem.l  (a7)+,a4/a5
                move.w   #%0000000110100000,$dff096 ;DMA sperren
                move.l   _GfxBase,a0     ; alte CopperList
                move.l   38(a0),$dff080  ; zurück schreiben
                clr.w    $dff088         ; und starten
                move.w   #%1000000110100000,$dff096 ; DMA wieder an
 #endasm
WriteBody(bmh)
 struct BitMapHeader *bmh;
 char *(*PlanePtr)[8]=&bm1.Planes[0];
 char **CurrentPtr,*ReadPointer,*pointer;
 unsigned short h=bmh->h,bpl=bmh->w>>3,d=bmh->Planes,PlaneOffset=0;
 unsigned short j,count;
 char n,x;
 #asm
                clr.l    d0
                clr.l    d1
                clr.l    d2
                move.l   _BodyChunk,a0  ;a0 = Zeiger auf Daten
                move.l   -4(a5),a1      ;Zeiger auf Planes[0]
                move.w   -18(a5),d0 ; d0 = Anzahl der Zeilen
                move.w   -20(a5),d1 ; d1 = Bytes pro Zeile
                move.w   -22(a5),d2 ; d2 = Anzahl der Planes
                clr.l    d4         ; d4=PlaneOffset
                clr.l    d6         ; d6=n=0
                clr.l    d5         ; PlaneOffset=0
                clr.l    d3         ;
                clr.l    d7
                bra      hstart
hloop:          move.w  d2,d3
                move.l  a1,a2      ;a2=CurrentPtr
                bra     pstart
ploop:          move.l  (a2)+,a3   ;a3=pointer
                add.l   d4,a3      ; += PlaneOffset
                clr.b   d5         ; count=0
cloop:          clr.w   d6
                move.b  (a0)+,d6   ;n=ReadPointer++
                tst.b   d6
                bmi     else
                addq.b  #1,d6      ;n++
                add.b   d6,d5      ;count += n
                bra     cpstart
cploop:         move.b  (a0)+,(a3)+
cpstart:        dbra    d6,cploop
                bra     cstart
else:           move.b  (a0)+,d7   ;x=ReadPointer++
                neg.b   d6         ;n = -n
                addq.b  #1,d6      ;n++
                add.b   d6,d5      ;count += n
                bra     xcstart
xcloop:         move.b  d7,(a3)+
xcstart:        dbra    d6,xcloop
cstart:         cmp.b   d1,d5
                bne.s   cloop
pstart:         dbra    d3,ploop
                add.w   d1,d4      ;PlaneOffset+=bpl
hstart:         dbra    d0,hloop
 #endasm
/*
 ReadPointer=BodyChunk;
 while (h--)
       j=d;
       CurrentPtr=PlanePtr;
       while (j--)
              pointer=*CurrentPtr++ + PlaneOffset;
              count=0;
              while (count != bpl)
                 if ((n = *ReadPointer++) >= 0)
                       n++;
                       count += n;
                       while (n--) *pointer++=*ReadPointer++;
                 else
                       x = *ReadPointer++;
                       n = -n;
                       n++;
                       count += n;
                       while (n--) *pointer++=x;
        PlaneOffset += bpl;
 */
CopyBitMap(s,d)
 struct BitMap *s,*d;
 char p=s->Depth,j=0;
 register unsigned long c;
 unsigned long size = s->BytesPerRow * s->Rows;
 register char *sp,*dp;
 while (p--)
       c=size;
       sp=s->Planes[j];
       dp=d->Planes[j++];
       while (c--) *dp++=*sp++;
FreePlanes(bm)
 register struct BitMap *bm;
 unsigned short p=bm->Depth,bpr=bm->BytesPerRow,r=bm->Rows;
 unsigned short planesize=bpr*r;
 char j=0;
 APTR PlanePtr;
 while (p--) if (PlanePtr=bm->Planes[j++]) FreeMem(PlanePtr,planesize);
CloseAnim(bm)
 struct DLTAChunk *Temp;
 FreeCopperList();
 FreePlanes(&bm1);
 FreePlanes(&bm2);
 if (BodyChunk) FreeMem(BodyChunk,BodySize);
 Current=Start;
 while (Current)
   Temp=Current->Next;
   FreeMem(Current,Current->Size);
   Current=Temp;
 Start=Current=Last=0;
MakeColorMap(ThisView)
 struct ViewPort *ThisView;
 char *c=ColorMap;
 unsigned short i,d=1<<BitMapHeader.Planes;
 unsigned short r,g,b;
 for (i=0;i<d;i++)
       r=*c++>>4;
       g=*c++>>4;
       b=*c++>>4;
       SetRGB4(ThisView,i,r,g,b);
BOOL InitRaster(bm,w,h,d)
```

Listing 4. »play.c« (Fortsetzung)

```
  struct BitMap *bm;
  unsigned short w,h,d;
  unsigned short planesize = (w>>3)*h;
  APTR PlanePtr;
  char j=0;
  InitBitMap(bm,d,w,h);
  while (d--)
          if (PlanePtr=AllocMem(planesize,MEMF_CHIP))
               bm->Planes[j++]=PlanePtr;
          else return(FALSE);
  return TRUE;
BOOL InitDisplay(bmh)
  register struct BitMapHeader *bmh;
  if (!InitRaster(&bm1,bmh->w,bmh->h,bmh->Planes)) return FALSE;
  if (!InitRaster(&bm2,bmh->w,bmh->h,bmh->Planes)) return FALSE;
  NewS1.ViewModes=NewS2.ViewModes=0;
  if (bmh->h > 256) NewS1.ViewModes=(NewS2.ViewModes |= LACE);
  if (bmh->w > 352) NewS1.ViewModes=(NewS2.ViewModes |= HIRES);
  NewS1.ViewModes = (NewS2.ViewModes |= VP_HIDE);
  NewS1.Width=bmh->w;
  NewS1.Height=bmh->h;
  NewS1.Depth=bmh->Planes;
  NewS2.Width=bmh->w;
  NewS2.Height=bmh->h;
  NewS2.Depth=bmh->Planes;
  return TRUE;
unsigned long ReadILBM(MyFile,End)
  unsigned long MyFile,End;
  unsigned long ID,Length,Offset;
  BOOL Error=FALSE;
  Offset=0;
  while (Offset < End && Error==FALSE )
    ID=NextLong;
    Length=NextLong;
    if (Length & 1) Length++;
    Offset+=Length+8;
    switch (ID)
      case BMHD:
      Read(MyFile,&BitMapHeader,sizeof(BitMapHeader));
      if (!InitDisplay(&BitMapHeader)) Error=TRUE;
      break;
      case CMAP:
      Read(MyFile,ColorMap,Length);
      break;
      case CAMG:
      NewS1.ViewModes=NewS2.ViewModes=NextLong | VP_HIDE;
      break;
      case BODY:
      BodySize=Length;
      if (!(BodyChunk=AllocMem(Length,MEMF_PUBLIC))) Error=TRUE;
      else Read(MyFile,BodyChunk,Length);
      break;
      case ANHD:
      Read(MyFile,&AnimHeader,Length);
      if (FirstTime)
           FirstTime=FALSE;
           DeltaMode=AnimHeader.operation;
           if (DeltaMode != SHORTDELTA && DeltaMode != VERTDELTA)
                printf("Unknown data copression...\n");
                Error=TRUE;
           InitMuluTable((BitMapHeader.w>>4)<<1);
      break;
      case DLTA:
      if (!(Current=AllocMem(Length+sizeof(struct DLTAChunk),MEMF_PUBLIC)))
           printf("Out of Memory!\n");
           Error=TRUE;
      else
           if (!Start) Start=Current;
           if (Last) Last->Next=Current;
           Last=Current;
           Read(MyFile,Current+1,Length);
           Current->Next=0;
           Current->Size=Length+sizeof(struct DLTAChunk);
      break;
      case GRAB:
      case DEST:
      case SPRT:
      case CCRT:
      case CRNG:
      printf("unused found.\n");
      Seek(MyFile,Length,0);
      break;
      default:
      printf("Sorry, but this is a corrupted IFF-File\n");
      Error=TRUE;
      break;
  return Error;
BOOL ReadIFF(MyFile)
  unsigned long MyFile;
  unsigned long ID,Length,Offset=0,End;
  BOOL Error=FALSE;
  if ((ID=NextLong) == FORM)
       End=NextLong;
       if ((ID=NextLong) != ANIM) Error=TRUE;
       else Offset+=4;
       while (Offset < End && Error==FALSE)
         if ((ID=NextLong) == FORM)
              Length=NextLong;
              if ((ID=NextLong) == ILBM)
                   Offset+=8;
                   if (ReadILBM(MyFile,Length-4)) Error=TRUE;
                   else Offset+=Length;
              else
                   printf("Sorry, but this is an mangled IFF-File...\n");
                   Error=TRUE;
         else Error=TRUE;
  else
       printf("This is not an IFF File.\n");
       Error=TRUE;
  return Error;
main(argc,argv)
  int argc;
  char **argv;
  unsigned long MyFile;
  char c;
  IntuitionBase=OpenLibrary("intuition.library",0);
  GfxBase=OpenLibrary("graphics.library",0);
  printf("\n*********************************************");
  printf("\n***          Delta Anim Player 0.8         ***");
  printf("\n*** written in Aztec C by: Christian Wolf ***");
  printf("\n***        Copyright (c) 1988 by CWF       ***");
  printf("\n*********************************************\n\n");
  argc--;
  if (!argc) printf("Need a filename as input...\n");
  argv++;
  if ((*argv)[0] == '-')
       argc--;
       switch (c=(*argv)[1])
                 case 'c':
                 case 'C':
                 Mode=CONT;
                 break;
                 case 'l':
                 case 'L':
                 Mode=LOOP;
                 break;
                 default:
                 printf("illegal option! -%c\n",c);
                 break;
       argv++;
  while(argc--)
       if (!(MyFile=Open(*argv++,MODE_OLDFILE)))
            printf("Can't open specified file...\n");
            continue;
       printf("Loading animation into memory... Please stand by.\n\n");
       if (!ReadIFF(MyFile))
            PlayAnim();
       CloseAnim();
       Close(MyFile);
  CloseLibrary(GfxBase);
  CloseLibrary(IntuitionBase);
```


Listing 4. »play.c« (Schluß)

Fonts mit allen Tricks

Der Amiga wird selbst mit vielen verschiedenen Zeichensätzen spielend fertig. Bisher gab es aber nur wenige Programme, mit denen Zeichensätze zu bearbeiten oder selbst zu erstellen sind. Mit dem »Fontdesigner« wird das anders.

Durch die Darstellung von Texten als Grafik stehen dem Amiga-Anwender zahlreiche Zeichentypen zur Wahl. Eine Vielzahl verschiedener Standardfonts steht bereit. Aber kreative Amiga-Benutzer möchten natürlich auch eigene Zeichensätze entwerfen oder bestehende ändern. Genau dieser Wunsch wird mit dem »Fontdesigner« Wirklichkeit. Bearbeiten Sie mit dem Fontdesigner alle Amiga-Zeichensätze, die eine Höhe von 50 Punkten nicht überschreiten. Sogar ein Programmiermodus ist enthalten, mit dem gezielt Zeichensätze oder auch Teile daraus geändert werden. In Tabelle 1 (Seite 53) finden Sie eine Beschreibung aller Fähigkeiten dieses Programms.

Der Fontdesigner ist in der Programmiersprache C geschrieben. Geben Sie die Listings 1 bis 6 mit dem Checksummer (Seite 159) ein. Nachdem die Listings alle einzeln auf einer Diskette gespeichert wurden, verwenden Sie die Hinweise in Tabelle 2a (für Aztec-Compiler V3.4a) beziehungsweise Tabelle 2b (für Lattice-Compiler), um das Programm zu compilieren und zu linken. Besitzen Sie die Programmservice-Diskette, entfällt diese Arbeit natürlich, die lauffähige Version befindet sich auf dieser.

Bevor Sie das Programm starten, noch einige Tips:

Der Fontdesigner benötigt den CLI-Befehl »Assign« beim Laden von Zeichensätzen.

Letzte Vorbereitungen

Falls Sie nur ein Laufwerk besitzen, sollten Sie den Fontdesigner in der Startup-Sequence starten und alle Fonts auf dieser Diskette verwalten, um sich häufiges Wechseln der Diskette zu ersparen. Außerdem sollten Sie in jedem Fall dafür sorgen, daß die Datei »Assign« im Verzeichnis »c« und die Datei »diskfont.library« im Verzeichnis »libs« auf Ihrer Bootdiskette zu finden sind. Besitzen Sie zwei Laufwerke, booten Sie einfach mit Ihrer Workbench; haben Sie nur ein Laufwerk, so sollten Sie die beiden genannten Dateien auf Ihre Fontdesigner-Diskette in die richtigen Verzeichnisse kopieren.

Nachdem Sie das Programm mit

```
Fontdesigner <RETURN>
```

vom CLI aus gestartet haben, gelangen Sie in den Hauptbildschirm (Bild 1). Von hier aus werden alle Funktionen des Programms kontrolliert; dies geschieht durch Anklicken des gewünschten »Schalters«. Der Hauptbildschirm ist — auch wenn er nicht so aussieht — ein ganz normales Amiga-Fenster, so daß mit Hilfe der Gadgets oben rechts andere Fenster nach vorne geklickt werden können, um das Multitasking des Amiga zu nutzen.

Den größten Teil des Bildschirms nimmt die Zeichenmatrix ein. Da sie 64 Kästchen breit und 50 Kästchen hoch ist, verarbeitet das Programm Fonts mit einer maximalen Größe von 64 mal 50 Punkten.

Um einen Punkt in der Matrix zu setzen, wird er einfach mit der linken Maustaste angeklickt, die rechte Maustaste dient zum Löschen bereits gesetzter Punkte. Sie können durch Überfahren der Matrix mit niedergehaltener linker Maustaste in diese sozusagen »hineinmalen«.

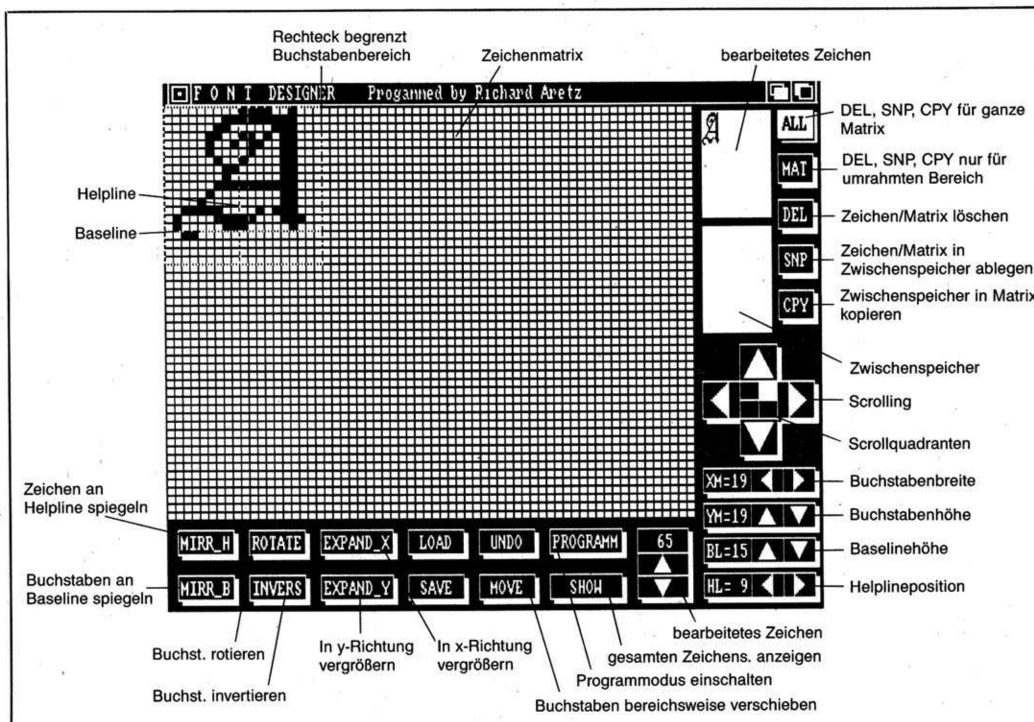


Bild 1. Der »Fontdesigner« gestattet kreative Veränderungen an den Zeichensätzen. Sogar die Programmierung diverser Änderungen ist vorgesehen.

Wichtig: Base- und Helpline

In der Matrix befinden sich ferner ein Rechteck und zwei Linien in oranger Farbe (alle Farbangaben gelten für eine Workbench, auf der die Bildschirmfarben der Original-Workbench-Diskette nicht verändert wurden). Das Rechteck umgrenzt das Feld, das beim Speichern eines Zeichensatzes auf die Diskette geschrieben wird — es gibt also die Größe der Zeichen an. Die waagerechte Linie ist die sogenannte Baseline, die bei der Gestaltung von Unterlängen (etwa beim »g«) wichtig ist (Bild 2). Die senkrechte Linie ist die »Helpline« — sie hat bei den Funktionen »spiegeln« und »scrollen« eine wichtige Bedeutung.

Das Feld neben dem »PROGRAMM«-Schalter dient zur Anwahl des Zeichens, das man bearbeiten will. Bevor eine Veränderung des Zeichensatzes durchgeführt werden kann, müssen natürlich erst einmal die vorhandenen Schriftarten (Fonts) von Diskette in den Arbeitsspeicher geladen werden. Verwenden Sie dazu den Schalter »LOAD«. Genaues hierzu finden Sie weiter unten.

Im obersten Feld steht der ASCII-Code des aktuellen Zeichens; klickt man dieses Feld an, kann ein neuer Code mit der Tastatur eingegeben werden. Mit den beiden Pfeilen wird die Zahl im obersten Feld hinauf- und heruntergezählt. Wenn Sie einen bestimmten Buchstaben bearbeiten wollen, tippen Sie ihn einfach per Tastatur ein.

Rechts neben der Zeichenmatrix befinden sich zwei kleine Fenster. Das obere Fenster stellt den in der Zeichenmatrix vergrößert dargestellten Buchstaben in Normalgröße dar. Im unteren Fenster ist der Inhalt eines Zwischenspeichers zu sehen, der von den »SNP«- und »CPY«-Funktionen verwendet wird.

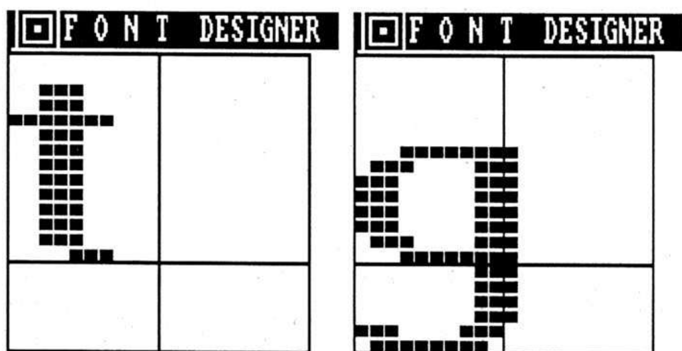


Bild 2. So muß die Baseline bei Unterlängen verwendet werden, damit das »g« richtig auf der Zeile sitzt

Oben rechts im Hauptdisplay befinden sich fünf Schalter mit den Aufschriften ALL, MAT, DEL, SNP und CPY.

ALL und **MAT**

Ist ALL aktiviert, gelten die unteren drei Befehle für die gesamte Zeichenmatrix; ist MAT eingeschaltet, gelten sie nur für den Bereich des gelben Rechtecks.

DEL

Der gerade bearbeitete Buchstabe wird gelöscht.

SNP

Kopiert den aktuellen Buchstaben in einen Zwischenspeicher.

CPY

Kopiert den Zwischenspeicher in die Matrix.

Unterhalb der gerade beschriebenen Schalter befindet sich das **Scrollfeld.** Es besteht aus den Richtungssymbolen und dem Quadrantenfeld in der Mitte. Die Helpline und die Baseline teilen die orange umrandete Buchstabenmatrix in vier Quadranten, die alle einzeln in alle Richtungen gescrollt werden können. Bevor sie also den Scroll-Vorgang mit einem Mausklick auf einen der Richtungspfeile starten, müssen Sie den gewünschten Quadranten im Quadrantenfeld anklicken. Vorsicht: Aus dem Quadranten »herausgescrollte« Punkte gehen verloren! Sie sind dann nur noch durch »UNDO« zu retten, wenn zwischendurch keine andere Operation durchgeführt wurde.

Die Schalter XM, YM, BL und HL

Mit diesen Schaltern wird die Größe des orangen Rechtecks sowie die Position von Help- und Baseline eingestellt. Mit **XM** ändert man die Höhe der Matrix; mit **YM** ihre Breite. **BL** und **HL** verschieben die Position der Base- beziehungsweise Helpline. Um die angezeigten Zahlenwerte zu ändern, klickt man mit der Maus auf den gewünschten Pfeil. Die Pfeilspitze zeigt dabei in die Richtung, in die sich die jeweilige Linie oder Rechteckbegrenzung verschiebt. Sollen die Zahlenwerte mit der Hand eingegeben werden, so klickt man auf die zu ändernde Zahl und gibt den neuen Wert mit der Tastatur ein.

Unterhalb der Zeichenmatrix befinden sich weitere Befehle des Programms, die wie gewohnt mit der Maus angeklickt werden. Sie beziehen sich nur auf das orange umrandete Rechteck.

MIRR_H

Der Buchstabe wird an der Helpline gespiegelt.

MIRR_B

Der Buchstabe wird an der Baseline gespiegelt.

Rotieren, spiegeln, vergrößern

ROTATE

Der ganze Buchstabe wird um 90 Grad nach links verdreht. Dazu werden innerhalb der Gelb umrandeten Matrix die Zeilen und Spalten vertauscht.

INVERT

Der Buchstabe wird invertiert.

EXPAND_X und **EXPAND_Y**

Der Buchstabe wird in X- beziehungsweise Y-Richtung vergrößert.

LOAD

Lädt einen Zeichensatz von der Diskette. Beim Anklicken von LOAD erscheint der LOAD-Requester (Bild 3), in dem der gewünschte Zeichensatz zum Laden ausgewählt wird. Das Programm lädt nun automatisch das Verzeichnis »DF0:fonts« und zeigt es auf dem Bildschirm an. Um einen Zeichensatz von einem anderen Diskettenlaufwerk zu laden, klicken Sie einfach auf »DF1:«, »DF2:« oder »DF3:«. Dann wird das »Fonts«-Verzeichnis des gewählten Laufwerks angezeigt. Mit Hilfe der beiden Pfeile wird ein Verzeichnis hinauf- und heruntergescrollt, falls es nicht vollständig in das Ausgabefenster paßt. Wollen Sie aus einem Unterverzeichnis in das übergeordnete (also eine Ebene höher liegendes) Verzeichnis springen, klicken Sie auf »PARENT«. Um beliebige Verzeichnis- und Device-Namen eingeben zu können, klickt man das Feld rechts neben »Drawer« mit der Maus an und gibt den gewünschten Namen per Tastatur ein.

Wie Sie sicher wissen, befinden sich im Verzeichnis »fonts« für jeden Zeichensatz ein Unterverzeichnis (Beispiel: »ruby«) und eine Steuerdatei (Beispiel: »ruby.font«). Der Fontdesigner stellt Unterverzeichnisse in Gelb und Dateien in Schwarz dar. Wichtig für das Laden von Zeichensätzen sind nur die Unterverzeichnisse! Wollen Sie also zum Beispiel von Ihrer Workbench-Diskette den Zeichensatz »ruby« laden, so klicken Sie auf das Unterverzeichnis »ruby«. Sie sehen dann in schwarzer Farbe die von diesem Zeichensatz vorhandenen Font-Höhen. Nach einem Klick auf die gewünschte Höhe und auf »OK« wird der ausgewählte Zeichensatz geladen. Eines sollten Sie allerdings beachten: Zeichensätze mit einer Höhe von mehr als 50 kann der Fontdesigner nicht bearbeiten.

Beschleunigtes Directory

Da der Amiga beim Laden von Verzeichnissen bekanntermaßen nicht gerade schnell ist, läßt sich mit einem Klick auf »FAST« ein Fast-Directory im momentan ausgewählten Verzeichnis erzeugen. Besonders interessant ist das natürlich für die »fonts«-Verzeichnisse Ihrer Disketten. Beim Speichern eines Zeichensatzes wird das Fast-Directory im Verzeichnis »fonts« daher auch automatisch auf den neuesten Stand gebracht. Verändern Sie andere Verzeichnisse, so ist das Fast-Directory mit einem weiteren Klick auf »FAST« zu aktualisieren.

Wenn Sie Ihre Zeichensätze nicht nach der auf dem Amiga üblichen Methode (also im Unterverzeichnis »fonts«) abspeichern, beachten Sie bitte die Hinweise in der Tabelle 3.

Sie müssen nicht immer den gesamten Zeichensatz laden. Geben Sie bei »FROM ASCII« den Bereichsanfang und bei »TO ASCII« das Bereichsende an. Die Eingaben dabei beziehen sich auf die Nummer des ASCII-Zeichens (65 entspricht beispielsweise dem großen »A«, 98 dem kleinen »a«).

DF0: DF1: DF2: DF3: PARENT
DRAWER DF0:fonts/sapphire
14
19
FROM ASCII: 65 TO ASCII:130 ALL
19 ASSIGN FAST
OK IO Error: none CANCEL

Bild 3. Der LOAD-Requester des Programms

Streng genommen läßt sich auch ein einzelnes Zeichen aus einem Zeichensatz laden. Der restliche Zeichensatz im Speicher wird dabei nicht gelöscht. Wählen Sie ALL, wird der gesamte Zeichensatz geladen.

Um den angewählten Zeichensatz zu laden, klickt man schließlich »OK« an; »CANCEL« dient zum Abbruch der Funktion.

SAVE

Zeichensatz abspeichern. Wie schon bei LOAD öffnet sich ein weiteres Fenster, in dem verschiedene Eingaben erfolgen können:

1. Der vollständige Font-Name, also der Pfadname, der zu diesem Font führt.
2. »HEIGHT« ist die Font-Höhe. Es wird, falls Sie die Höhe nicht ändern, automatisch der Wert für YM (Y-Matrix) benutzt.
3. »LO CHAR« und »HIGH CHAR«: Nicht jeder Zeichensatz muß alle ASCII-Codes ausfüllen. In diesen beiden Feldern steht der erste »LO« beziehungsweise der letzte »HIGH«-ASCII-Code, der zu diesem Font gehört. Verändern Sie diese Werte — falls nötig — durch einfaches Anklicken mit der Maus und anschließender Eingabe des Wertes über die Tastatur.

Proportional oder nicht?

4. »PROPORTIONAL« und »FIXED WIDTH«: Bei einem Proportionalzeichensatz kann jeder Buchstabe eine individuelle Breite besitzen (sie wird vom Programm beim Speichern automatisch berechnet); »Fixed-Width«-Zeichensätze haben dagegen bei allen Zeichen die Breite, die bei »XM« eingestellt wurde. Wählen Sie also mit der Maus, wie Ihr Zeichensatz abgespeichert werden soll.
5. »OK«: Zeichensatz speichern. Da der Speichervorgang vor allem bei größeren Zeichensätzen einige Minuten dauern kann, werden während des Speicherns in der linken oberen Ecke des Bildschirms einige Zwischenmeldungen angezeigt:

»Writing xxxx.font«: Die ».font«-Datei für den jeweiligen Zeichensatz wird erzeugt.

»Examining Font«: Interne Berechnungen werden durchgeführt.

»Writing char data«: Der eigentliche Zeichensatz wird geschrieben.

»Creating new Fast-Dir«: Das Fast-Directory wird (falls eines vorhanden war) aktualisiert.

Falls Sie die Meldung »Error! Font not saved!« bekommen sollten, haben Sie wahrscheinlich versucht, einen Zeichensatz in ein nicht existierendes Verzeichnis zu speichern.

6. »GET NAME«: Name wie unter LOAD beschrieben auswählen.
7. »CANCEL«: Zeichensatz nicht speichern.

Mehr Komfort mit UNDO

UNDO

Dieser Befehl macht die letzte Operation rückgängig.

MOVE

Verschiebung von Zeichen innerhalb eines Zeichensatzes. Sie müssen bei MOVE drei Zahlenwerte in das geöffnete Fenster eingeben:

1. Bei »FROM ASCII«: den Anfang des Bereichs, der verschoben werden soll.
2. Bei »UNTIL ASCII«: das Ende dieses Bereichs.
3. Bei »TO ASCII«: den Anfang des Bereichs, an den verschoben werden soll.

Beispiel: Sie möchten die Zeichen mit den ASCII-Codes 65 bis 90 nach 128 verschieben. Dann ist FROM ASCII-65; UNTIL ASCII=90; TO ASCII=128.

PROGRAMM

Dies ist eine der interessantesten Funktionen von »Font-Designer«! Möchten Sie bei einem Zeichensatz mehrere ASCII-Codes auf die gleiche Art und Weise verändern, können Sie das getrost dem Amiga überlassen. Geben Sie dazu in den entsprechenden Feldern den ersten (»START«) und letzten ASCII-Code (»END«) sowie die Schrittweite (»STEP«) an. Nun werden die einzelnen Funktionen ausgewählt, die der Amiga mit dem gewählten Zeichenbereich durchführen soll.

Die Aufschrift der Schalter entspricht in etwa der Aufschrift im Hauptdisplay (Bild 4). »Q1« bis »Q4« bezeichnen die Quadrantenauswahl für das Scrollen. Dabei steht »Q1« für oben links, »Q2« für oben rechts, »Q3« für unten links und »Q4« für unten rechts. Das »Programm« wird einfach mit Hilfe von Mausklicks auf die gewünschten Felder eingegeben. Sollten Sie sich »vertippt« haben, so läßt sich das eingegebene Programm durch einen Klick auf »PZ« löschen. Schließlich wird der Vorgang mit »GO« gestartet — mit diesem Schalter kann das Programm auch wieder abgebrochen werden. Rechts unten befindet sich der PZ (Programmzähler), in dem der zuletzt eingegebene Programmschritt angezeigt wird.

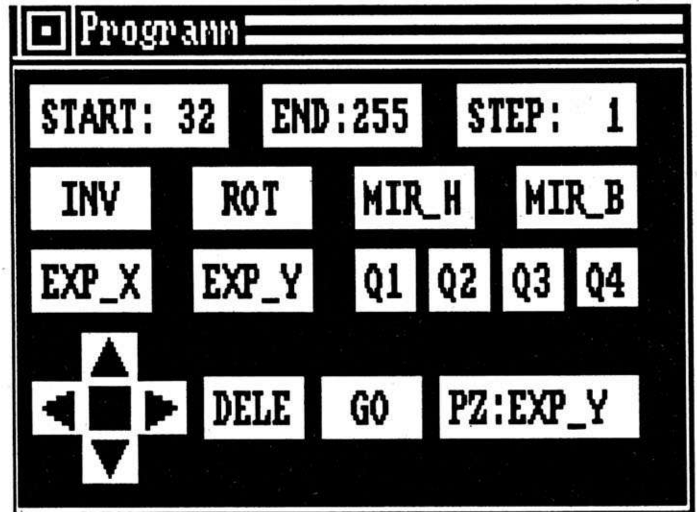


Bild 4. Durch Auswahl im »PROGRAMM«-Fenster wird vieles automatisch erledigt.

Die Programm-Automatik

Ein Beispiel: Sie wollen sämtliche kleinen Buchstaben ihres Zeichensatzes nach links drehen. Dann geben Sie nach dem Klick auf »PROGRAMM« bei »START« 97 und bei »END« 123 ein. Nun klicken Sie einmal auf »ROT«. Daraufhin erscheint bei »PZ« der gerade eingegebene Programmschritt »ROT«. Nun starten Sie das Ganze mit »GO«. Rechts unten neben dem Programmfenster wird der ASCII-Code des gerade bearbeiteten Zeichens angezeigt.

Verlassen Sie dieses Fenster durch Betätigen des Schließ-Gadgets oben links.

SHOW

Zeigen des gesamten Zeichensatzes. Von jedem Zeichen wird der durch das orange Rechteck umgrenzte Bereich in Originalgröße angezeigt. Sollten alle Zeichen nicht auf einmal auf den Bildschirm passen, wird mit der rechten Maustaste weitergeschaltet. Die linke Maustaste beendet die Ausgabe.

Um sich mit den vielen Fähigkeiten des Fontdesigners vertraut zu machen, brauchen Sie einfach ein bißchen Übung. Ein Vorschlag: Versuchen Sie doch einmal, Ihre Lieblingszeichensätze mit deutschen Sonderzeichen zu versehen. *(Richard Aretz/ Andreas Lietz/rs)*

- Zeichensätze sind bis zu einer Größe von 64 x 50 Punkten zu bearbeiten.
- Das angewählte Zeichen wird in einer Matrix vergrößert und in Originalgröße dargestellt. Um es zu verändern, »malt« man mit der Maus in die Matrix.
- Ein Zeichen ist in einen Zwischenspeicher zu kopieren, um es an anderer Stelle zu verwenden.
- Zeichen invertieren, spiegeln, drehen, vergrößern, scrollen.
- Mit UNDO Befehle rückgängig machen.
- Mit Hilfe eines Programm-Modus diese Veränderungen bei vielen Zeichen automatisch durchführen.
- Alle Zeichen auf einmal in Originalgröße anzeigen.
- Zeichen innerhalb eines Zeichensatzes bereichsweise verschieben.
- Standard-Amiga-Zeichensätze laden und speichern.
- Zeichensätze mit Proportionalschrift oder gleichbleibender Breite speichern.

Tabelle 1. Was kann der Fontdesigner? Die Kurzübersicht zeigt alle Vorzüge des Programms.

```
COMPILIEREN:
cc h0.c +l -s
cc h1.c +l -s
cc h2.c +l -s
cc h3.c +l -s
cc fontdesigner +l -s
LINKEN:
ln fontdesigner.o h0.o h1.o h2.o h3.o -lm32 -lc32
```

Tabelle 2a. Die Compiler- und Linker-Anweisungen für den Aztek-C-Compiler, Version 3.4a

```
COMPILIEREN:
lc h0
lc h1
lc h2
lc h3
lc fontdesigner
LINKEN:
blink LIB:c.o + fontdesigner.o + h0.o + h1.o + h2.o +
h3.o LIB LIB:amiga.lib + LIB: m.lib + LIB: c.lib
```

Tabelle 2b. Die Anweisungen für den Lattice-C-Compiler

Unter normalen Umständen ist es das beste, wenn Sie Ihre Zeichensätze immer in einem Verzeichnis mit dem Namen »fonts« verwalten. Sollten Sie sie aber in ein anderes Verzeichnis speichern wollen (z.B. bei Verwendung einer Festplatte), dann beachten Sie bitte, daß der Amiga beim Laden von Diskettenfonts den Befehl »Assign fonts: newdir:« benötigt, wobei »newdir« für den Namen Ihres Fontverzeichnisses steht. Der »Assign«-Befehl wird normalerweise von Fontdesigner automatisch bei der Auswahl eines anderen Diskettenlaufwerks für das Verzeichnis »fonts« dieser Diskette ausgeführt.

Um den Befehl auf Ihr Zeichensatzverzeichnis manuell anzuwenden, klicken Sie im LOAD-Requester einfach auf »ASSIGN«, sobald Sie Ihr Zeichensatzverzeichnis ausgewählt haben. Dann können Sie den gewünschten Font-Namen anklicken, sich in dem nun erscheinenden neuen Verzeichnis eine Font-Höhe aussuchen und den Zeichensatz laden.

Falls Sie den Klick auf »ASSIGN« vergessen sollten, erscheint die Meldung »I can't find this font«, und der Zeichensatz »topaz 8« wird automatisch geladen. Sollte Ihnen das passieren, klicken Sie im LOAD-Requester auf »PARENT«, anschließend auf »ASSIGN« und dann auf den Namen Ihres Zeichensatzes. Nach dem Klick auf »OK« sollten nun keine Probleme mehr auftreten.

Tabelle 3. Wichtige Hilfestellung bei Problemen mit dem Fontdesigner

Programmname:	»Fontdesigner«
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.4a / Lattice-C
Aufrufe:	siehe Tabelle 2a bzw. 2b
Bemerkung:	siehe Tabelle 3 sowie Text

```
Programm : FontDesigner

-----------------------

HyO /*---(FontDesigner)------- File test.c (Hauptprogramm) -----
    -----*/
H6  #include "header.h"
82  /*------------------------ Globale Variablen ---------------
    -----------*/
X0  struct IntuitionBase *IntuitionBase;
rU  struct GfxBase *GfxBase;
c1  ULONG DiskfontBase,DosBase;
K9  long XMatrix=8,YMatrix=8,BaseLine=7,HelpLine=4,DMode=1,Ascii
    =65;
Kd  long Quadrant=1,undoascii=65;
Az  struct TextFont *tftopaz,*tfuser;
I4  struct Window *wind;
ht  struct RastPort *rast;
xK  ULONG *cmatrix;
wt  BYTE *drawmem;
h6  struct Image *oima,*drima,*mima;
P1  struct FileInfoBlock *f_info;
qh  ULONG maske[]=
Dg  {
mC3    0x80000000,1073741824,536870912,268435456,134217728,67108
       864,
Gr     33554432,16777216,8388608,4194304,2097152,1048576,524288,

I7     262144,131072,65536,32768,16384,8192,4096,2048,1024,512,2
       56,128,
r2     64,32,16,8,4,2,1,
fb0 };
Gr  ULONG cmemo[100],scmemo[100],undomem[100];
ev  extern short recdat[][4],FAscii,TAscii;
```

```
jV  /*-------------------- H A U P T P R O G R A M M -----------
    --------*/
7T  main()
Nq  {
DX  ULONG class;
LY  USHORT code;
jB  short mox,moy,i,mode,j;
uK  long *adr,*point;
CT  BYTE flag=0;
vA3    Oplibs();
nB     if(GetMemory(1)<0)
Vy     {
Qf6       Shure(3);
PO        Cllibs();
3R        exit(0);
d83    }
qi     if(Display()<0)
b4     {
W16       Shure(3);
VU        Cllibs();
HF        GetMemory(0);
AY        exit(0);
kF3    }
Of     tfuser=tftopaz=rast->Font;
K1     FOREVER
jC     {
Hn6       Wait(1<<wind->UserPort->mp_SigBit);
SX        while(ExaIntui(wind,&class,&code))
mF        {
E89          mox=wind->MouseX;moy=wind->MouseY;
Wo           switch (class)
pI           {
KWC             case CLOSEWINDOW:
LmF                if(Shure(0)==1)break;
bS                 if(flag==1){ RemFont(tfuser);CloseFont(tfuser
                   ); }
JL                 CloseWindow(wind);
XV                 GetMemory(0);
nm                 Cllibs();
Rp                 exit(0);
F4C             case VANILLAKEY:
IkF                adr=cmatrix+100*Ascii;point=undomem;
Aj                 undoascii=Ascii;
ZH                 for(j=0;j<=99;j++,point++,adr++)*point=*adr;

k4                 Ascii=code;
```

```
tm              AsciiOut();
CQ              Mem_to_Matrix();
3C              break;
bzC          case MOUSEBUTTONS:
wu           if(code!=SELECTDOWN && code!=MENUDOWN)break;
6f           mode=1;
d9           if(code==MENUDOWN)mode=0;
X5           i=Prurec(mox,moy,recdat,0,40);
3t           if(i<0)break;
kI           if(i>=4 && i<=15){ ChangeMatrix(i);break; }
Pn           if(i != 32)
Dg           {
YOF             adr=cmatrix+100*Ascii;point=undomem;
Qz              undoascii=Ascii;
pX              for(j=0;j<=99;j++,point++,adr++)*point=*adr;

LqC          }
mh           if(i>=29)Invert(rast,recdat[i]);
pf           switch (i)
Kn           {
WpF             case 0: DrawInto(mode);break;
XV              case 1: case 2: case 3:
nmI                ChangeAscii(i-1);
NW                 break;
zcF             case 16:
RgI                if(DMode==1)break;
b0                 DMode=1;
79                 Invert(rast,recdat[16]);Invert(rast,recdat
                   [17]);
Sb                 break;
71F             case 17:
UiI                if(DMode==0)break;
d1                 DMode=0;
CE                 Invert(rast,recdat[16]);Invert(rast,recdat
                   [17]);
Xg                 break;
Y8F             case 18: case 19: case 20:
oPI                Invert(rast,recdat[i]);
jq                 Delete(i-18);
xy                 Invert (rast,recdat[i]);
cl                 break;
bXF             case 21: case 22: case 23: case 24:
tUI                Invert(rast,recdat[i]);
ik                 Scroll(i-21);
vW                 Invert(rast,recdat[i]);
hq                 break;
iaF             case 25: case 26: case 27: case 28:
xlI                if((i-25)==Quadrant)break;
c8                 Invert(rast,recdat[Quadrant+25]);
fN                 Quadrant=i-25;
1c                 Invert(rast,recdat[i]);
nw                 break;
0xF             case 29: case 30: Mirror(i-29);break;
Qe              case 31: Rotate();break;
nX              case 32: Undo();break;
m2              case 33: Revers();break;
R4              case 34:
dUI                if(flag==1){ RemFont(tfuser);CloseFont(tfu
                   ser); }
Vt                 flag=1;
7U                 j=LoadDiskFont();
Zf                 if(j==-3){ flag=0;Shure(3);break; }
pt                 if(j==-2){ flag=0;break; }
TK                 if(j==-1)
OT                 {
1gL                   tfuser=tftopaz;
q3                    Shure(1);
by                    flag=0;
8dI                }
MG                 if(flag==1)Shure(6);
17                 else Shure(7);
xI                 for(j=FAscii;j<=TAscii;j++)
2ML                   Font_to_Mem(j);
CUI                Shure(6);
78                 Ascii=65;
Sb                 XMatrix=tfuser->tf_XSize;
xI                 if(XMatrix > 64)XMatrix=64;
ci                 YMatrix=tfuser->tf_YSize;
Q2                 if(YMatrix > 50)YMatrix=50;
J1                 BaseLine=tfuser->tf_Baseline;
Ff                 if(BaseLine > 50)BaseLine=49;
WU                 HelpLine=XMatrix/2;
Rf                 Mem_to_Matrix();
cn                    AsciiOut();Matout();
JS                    break;
nAF                case 35: case 36:
1rI                   if(i==35) Expand_x();
2g                    else Expand_y();
NW                    break;
oYF                case 37: SaveFont();break;
kN                 case 38: Programm();break;
NZ                 case 39: Show();break;
uu                 case 40: MoveAscii();break;
X2C             }
yt              if(i>=29)Invert(rast,recdat[i]);
Z49          }
a56       }
b63    }
c70 }
(C) 1987 M&T
```

Listing 1. Das Programm »fontdesigner« ist in der Sprache C geschrieben. Geben Sie das Listing bitte mit dem Checksummer (Seite 159) ein. Beachten Sie bitte die Compiler- und Linker-Anweisungen in Tabelle 2a beziehungsweise 2b.

Programmname: h0.c

Computer: s. Listing 1

```
Programmname: h0.c
------------------

LJ0  /* --(FontDesigner)-------- File h0.c ----------- */
9E   #include <exec/memory.h>
Rg   #include <intuition/intuition.h>
L9   #include <intuition/intuitionbase.h>
3q   #include <libraries/dos.h>
i5   #include <graphics/gfxmacros.h>
MB   #include "header.h"
aN   /*-------------------- Externe Variablen aus main --------
     ----------------*/
TS   extern struct TextFont *tfuser,*tftopaz;
48   extern short *drawmem;
18   extern long XMatrix,YMatrix,BaseLine,Ascii;
Az   extern ULONG *cmatrix;
nb   extern struct FileInfoBlock *f_info;
f1   extern struct RastPort *rast;
JL   extern struct Image *oima,*mima;
GY   /*---------------------- Daten  für GetFilename()  -------
     ----------------*/
Xo   short fdat[][4]=
IB3     { 10,16,48,28 },
9b      { 69,16,107,28 },
yi      { 128,16,166,28 },
I3      { 187,16,225,28 },
2X      { 248,16,302,28 },
Mb      { 70,33,302,45 },
kC      { 10,50,272,152 },
Bg      { 282,50,302,60 },
JL      { 282,142,302,153 },
PO      { 10,174,194,186 },
gA      { 10,191,64,203 },
DI      { 248,191,302,203 },
Q3      { 264,174,302,186 },
E1      { 202,174,256,186 },
Sa      { 10,157,128,169 },
4P      { 149,157,251,169 },
ZN      { 272,157,302,169 },
c8      { 97,191,215,203 },
tp0  };
8n   char *ftxt[]=
cc3     "DF0:","DF1:","DF2:","DF3:","PARENT","DRAWER","","","",
        "","  OK",
NH      "CANCEL","FAST","ASSIGN","FROM ASCII:","TO ASCII:","ALL
        ",
dp      "IO Error: none",
yu0  };
ai   short fpfeil[][7]=
uI3     { 292,51,284,59,300,59,-2 },
```

Listing 2. »h0.c« bitte mit dem Checksummer eingeben

```
    { 292,151,284,143,300,143,-2 },
};
struct StringInfo gf_info1,gf_info2;
struct Gadget gf_gad2=
    NULL,14,177,178,9,GADGHCOMP,NULL,STRGADGET,
    NULL,NULL,NULL,NULL,&gf_info2,1,NULL,
};
struct Gadget gf_gad1=
    &gf_gad2,74,36,228,9,GADGHCOMP,RELVERIFY,STRGADGET,
    NULL,NULL,NULL,NULL,&gf_info1,1,NULL,
};
char edrawer[100]="DF0:fonts";
char efname[100];
short FAscii=0,TAscii=255;
/*----------------------- Daten für SaveFont() ---------------------------*/
unsigned long ublock1[]=
    0x3f3,0,1,0,0,0x35e,0x3e9,0x35e,0x70004e75,0,0,0x0c000000,
    0x001a0f80,0x00010000,0,0,0,0,0,0,0,0,0,0xc00,0x1a,0,
    0,0x0062000f,0x00060001,0,0x6e,0x66e0000,0x9f20000,0xbb40000
};
unsigned long ublock2[]=
    0,0x03ec0000,0x00060000,0x0,0x000e0000,0x00440000,0x005c0000,
    0x00620000,0x00660000,0x006a0000,0,0x03f20000,
};
short sdat[][4]=
    { 10,16,64,28 },
    { 74,16,300,28 },
    { 10,33,88,45 },
    { 88,33,190,45 },
    { 190,33,300,45 },
    { 10,50,112,62 },
    { 206,50,300,62 },
    { 10,67,64,79 },
    { 246,67,300,79 },
    { 120,67,190,79 },
};
char *stxt[]=
    "DRAWER","","HIGHT:","LOW CHAR:","HIGH CHAR:","PROPORTIONAL",
    "FIXED WIDTH"," OK","CANCEL","GET NAME",
};
struct StringInfo s_info;
struct Gadget s_gad=
    NULL,78,19,218,9,GADGHCOMP,NULL,STRGADGET,
    NULL,NULL,NULL,NULL,&s_info,1,NULL,
};
char fontbreit[256];
/*h0.1----------------------- Diskfont laden ----------------------------*/
BYTE LoadDiskFont()
struct TextAttr txtatr;
BYTE erfolg,hmem[100];
long i;
double atof();
static BYTE count=0;
    if(count==0)
        strcpy(efname,"");
        strcpy(edrawer,"DF0:fonts");
        count++;
    erfolg=GetFilename(edrawer,efname);
    if(erfolg < 0) return(erfolg);
    strcpy(hmem,edrawer);
    strcpy(edrawer+strlen(edrawer),".font");
    txtatr.ta_Name=edrawer;
    txtatr.ta_YSize=(UWORD)atof(efname);
    txtatr.ta_Style=0;
    txtatr.ta_Flags=0;
    tfuser=OpenDiskFont(&txtatr);
    i=0;
    if(tfuser==0)
        tfuser=tftopaz;
        i=-1;
    edrawer[strlen(edrawer)-5]=0;
    return(i);
/*h0.2----------------------- Filename Holen ----------------------------*/
BYTE GetFilename(drawer,filename)
char filename[],drawer[];
ULONG flags=WINDOWDRAG| WINDOWCLOSE| WINDOWDEPTH| SMART_REFRESH|
        ACTIVATE;
ULONG iflags=MOUSEBUTTONS| CLOSEWINDOW| GADGETUP| GADGETDOWN|
DISKINSERTED;
struct Window *wn;
struct RastPort *rp;
BYTE i,offs=3,*names,*point,flag,mode=0;
short mox,moy;
ULONG class;
USHORT code;
short fzahl,start=1,nr,aktdrive=0;
short DirScroll();
  wn=Make_Window(20,10,320,209,"GET FILENAME",flags,iflags,0);
  if(wn==0) return(-3);
  names=AllocMem(1024,MEMF_CLEAR);
  if(names==0) return(-3);
  rp=wn->RPort;
  SetAPen(rp,1);RectFill(rp,0,11,320,209);
  Box(rp,2,12,317,207,0,JAM1);
  for(i=0;i<=17;i++)
     SetAPen(rp,0);
     RectFill(rp,fdat[i][0],fdat[i][1],fdat[i][2],fdat[i][3]);
     Box(rp,fdat[i][0],fdat[i][1],fdat[i][2],fdat[i][3],2,JAM1);
     Box(rp,fdat[i][0]-1,fdat[i][1],fdat[i][2]+1,fdat[i][3],2,JAM1);
     if(i==5)
        offs=-57;SetAPen(rp,0);
        RectFill(rp,10,33,64,45);
     else offs=3;
     Text_out(rp,fdat[i][0]+offs,fdat[i][1]+3,3,0,JAM2,ftxt[i],0);
  FDrawPolygon(rp,fpfeil[0],3,JAM2);
  FDrawPolygon(rp,fpfeil[1],3,JAM2);
  gf_info1.Buffer=drawer;gf_info2.Buffer=filename;
  gf_info1.UndoBuffer=NULL;gf_info2.UndoBuffer=NULL;
  gf_info1.MaxChars=100;gf_info2.MaxChars=100;
  AddGadget(wn,&gf_gad1,-1);AddGadget(wn,&gf_gad2,-1);
  RefreshGadgets(&gf_gad1,wn,NULL);
  fzahl=GetDir(names,drawer,mode);
  DirOut(rp,names,start,fzahl);
  FOREVER
     while (ExaIntui(wn,&class,&code))
        mox=wn->MouseX;moy=wn->MouseY;
        switch (class)
           case CLOSEWINDOW:
              flag=-2;
           ende:
              FreeMem(names,1024);
              RemoveGadget(wn,&gf_gad1);RemoveGadget(wn,&gf_gad2);
              CloseWindow(wn);
              return(flag);
           case DISKINSERTED:
              strcpy(drawer,ftxt[aktdrive]);
              strcpy(&drawer[4],"fonts");
              Assign(drawer);
           case GADGETUP: case GADGETDOWN:
              RefreshGadgets(&gf_gad1,wn,NULL);
              fzahl=GetDir(names,drawer,mode);
              start=1;
              DirOut(rp,names,start,fzahl);
              break;
           case MOUSEBUTTONS:
              if(code != SELECTDOWN)break;
              i=Prurec(mox,moy,fdat,0,16);
              if(i<0)break;
              if(i!=6 && i<14)Invert(rp,fdat[i]);
              switch(i)
                 case 0: case 1: case 2: case 3: case 4:
                 case 12:
                    if(i!=4 && i!=12)
                       aktdrive=i;
                       strcpy(drawer,ftxt[i]);
                       strcpy(&drawer[4],"fonts");
                       Assign(drawer);
                    else
                       if(i==4)
                         Parent(drawer);
                    if(i==12)mode=1;
                    fzahl=GetDir(names,drawer,mode);
```

```
1Z                    mode=0;
K9                    start=1;
V2                    DirOut(rp,names,start,fzahl);
cd                    Invert(rp,fdat[i]);
7G                    break;
PlI                case 6:
pQL                   nr=(moy-52)/10+start;
OF                    if(nr>=fzahl)break;
RC                    i=52+(nr-start)*10;
vq                    SetDrMd(rp,COMPLEMENT);
9U                    RectFill(rp,11,i,271,i+7);
mj                    Delay(5);RectFill(rp,11,i,271,i+7);
dY                    point=names;
5K                    for(i=1;i<=nr-1;i++)
WtO                      point++;
cB                       point+=strlen(point)+1;
gKL                   moy=*point;point++;
uY                    if(moy==1)
ToO                      if(drawer[strlen(drawer)-1] != ':')
yKR                         strcpy(drawer+strlen(drawer),"/\0");
qJO                      strcpy(drawer+strlen(drawer),point);
zq                       fzahl=GetDir(names,drawer,mode);
fU                       start=1;
qN                       DirOut(rp,names,start,fzahl);
I2L                   else strcpy(filename,point);
Sb                    break;
wMI                case 7: case 8:
aEL                   start=DirScroll(wn,names,start,fzahl,i);
12                    Invert(rp,fdat[i]);
Wf                    break;
LeI                case 10: case 11:
aqL                   if(i==10) flag=0;
AZ                    else flag=-2;
ef                    Delay(5);Invert(rp,fdat[i]);
92                    goto ende;
4eI                case 13:
FML                   ModifyIDCMP(wn,wn->IDCMPFlags&(**ßDISKINSERTED));
tY                    Assign(drawer);
HN                    ModifyIDCMP(wn,wn->IDCMPFlags| DISKINSERTED);
CD                    Invert(rp,fdat[i]);
hq                    break;
DoI                case 14:
VIL                   SetAPen(rp,3);SetDrMd(rp,JAM2);
Z8                    FAscii=(short)input_zahl(wn,fdat[14][0]+91,
Plk                                                  fdat[14][1]+3,3,"");
awL                   if(FAscii<0 || FAscii>TAscii)FAscii=0;
nw                    break;
MyI                case 15:
bOL                   SetAPen(rp,3);SetDrMd(rp,JAM2);
L6                    TAscii=(short)input_zahl(wn,fdat[15][0]+75,
Yvn                                                  fdat[15][1]+3,3,"");
vML                   if(TAscii>255 || TAscii<FAscii)TAscii=255;
t2                    break;
V8I                case 16:
RSL                   Invert(rp,fdat[i]);
UR                    FAscii=0;TAscii=255;
Kb                    Delay(5);
Jb                    Invert(rp,fdat[16]);
z8                    break;
1TF             RefreshGadgets(&gf_gad1,wn,NULL);
9c9          ErrOut(rp);
qb0 /*--- Unterfunktion zu h0.2 ---*/
45  short DirScroll(wn,names,start,fzahl,mode)
8o  struct Window *wn;
mQ  char *names;
3U  short start,fzahl,mode;
xH  ULONG class;
5I  USHORT code;
wE3    Delay(10);
nU     FOREVER
vf6       switch (mode)
Vs9          case 7:
DpC             if(start>1)
mHF                start--;
1K                 DirOut(wn->RPort,names,start,fzahl);
GPC             break;
c09          case 8:
o1C             if(start<(fzahl-10))
h8F                start++;
6P                 DirOut(wn->RPort,names,start,fzahl);
LUC             break;
Um6       if(ExaIntui(wn,&class,&code))break;
RS3    return(start);
VZO /*h0.3--------------------- Direktory laden ------------------------------*/
LR  short GetDir(namen,dirname,mode)
ZS  BYTE mode;
oX  char *namen,*dirname;
hc  struct FileLock *lockp;
Tc  register short anz;
qm  short flag;
Ln  char *point=namen;
p93    if((lockp=Lock(dirname,ACCESS_READ))==0)
fG6       return(-1);
he3    if(Examine(lockp,f_info)==0)return(-1);
g4     if(f_info->fib_DirEntryType<0) return(-1);
Kd     flag=FastDir(0,dirname,point,anz);
co     if(flag>0 && mode==0)
S16       SortDir(point,flag);
DM        return(flag);
y23    while(ExNext(lockp,f_info)==-1)
8o6       if(f_info->fib_DirEntryType<0)
4k9          *namen=0;
b06       else
Ar9          *namen=1;
Bb6       namen++;
Ym        strcpy(namen,f_info->fib_FileName);
5v        namen+=(strlen(f_info->fib_FileName)+1);
M1        anz++;
f23    UnLock(lockp);
j2     if(IoErr()==ERROR_NO_MORE_ENTRIES)
1U6       if(mode==1) FastDir(1,dirname,point,anz);
Jf        SortDir(point,anz);
pA        return(anz);
pD3    else return(-1);
FsO /*h0.4-------------------- Direktory ausgeben -----------------------------*/
JB  void DirOut(rp,names,start,max)
cG  char *names;
JZ  struct RastPort *rp;
XG  short start,max;
zr  short xa=13,ya=52,i,anz=max-start;
5F  char farbe;
YH3    SetAPen(rp,0);SetDrMd(rp,JAM1);
F3     RectFill(rp,11,51,271,151);
zf     for(i=1;i<=start-1;i++)
z56       names++;
y4        names+=strlen(names)+1;
JT3    if(anz>10)anz=10;
ET     for(i=1;i<=anz;i++,ya+=10)
Rk6       farbe=2;
wG        if(*names==1)
Wq9          farbe=3;
6C6       names++;
3S        Text_out(rp,xa,ya,farbe,0,JAM2,names,0);
6C        names+=strlen(names)+1;
eq0 /*h0.5------------------- Io Fehler ausgeben ----------------------------*/
HF  void ErrOut(rp)
cs  struct RastPort *rp;
28  long err;
2A  char htxt[15];
mw3    err=IoErr();
Gj     if(err<103 || err>231)
tB6       strcpy(htxt,ftxt[17]);
F23    else
oY6       sprintf(htxt,"IO Error:  %3.0f",(float)err);
a13    Text_out(rp,fdat[17][0]+3,fdat[17][1]+3,3,0,JAM2,htxt,14);
6G     sprintf(htxt,"%3.0f",(float)FAscii);
```

Listing 2. (Fortsetzung)

```
wI       Text_out(rp,fdat[14][0]+91,fdat[14][1]+3,3,0,JAM2,htxt,0);
Ge       sprintf(htxt,"%3.0f",(float)TAscii);
4e       Text_out(rp,fdat[15][0]+75,fdat[15][1]+3,3,0,JAM2,htxt,0);
q50   /*h0.6-------------- Fast Directory lesen oder erzeugen -----------------*/
Mc    short FastDir(mode,drawer,nam,anz)
tx    char mode,*drawer,*nam;
dK    short anz;
Y1    ULONG datei;
Xf    char dateiname[80],text[10],*point=nam;
26    short bytes=0,i;
Cp3      strcpy(dateiname,drawer);
1J       if(drawer[strlen(drawer)-1]==':')
Y76         strcpy(text,"fd.fdir");
WJ3      else
Ur6         strcpy(text,"/fd.fdir");
J93      strcpy(dateiname+strlen(dateiname),text);
tW       switch(mode)
116         case 0:
2q9            datei=Open(dateiname,MODE_OLDFILE);
28             if(datei==0)return(-1);
6G             if(Read(datei,&anz,2)!=2) goto abbruch;
Bb             Read(datei,nam,1024);
1u             break;
tA6         case 1:
U49            for(i=1;i<=anz-1;i++)
3QC               point++;
7j                bytes+=strlen(point)+2;
Aj                point+=strlen(point)+1;
ED9            datei=Open(dateiname,MODE_NEWFILE);
CI             if(datei==0)return(-1);
dH             if(Write(datei,&anz,2)!=2) goto abbruch;
aS             if(Write(datei,nam,bytes)!=bytes) goto abbruch;
v4             break;
1u3      Close(datei);
vG       return(anz);
1RO   abbruch:
4x3      Close(datei);
8j       return(-1);
qp0   /*h0.7---------------- Textfont auf Diskette schreiben --------------------*/
k2    void SaveFont()
ec    ULONG flags=ACTIVATE|WINDOWCLOSE|WINDOWDRAG;
TA    ULONG iflags=MOUSEBUTTONS|CLOSEWINDOW;
vF    ULONG class;
3G    USHORT code;
e0    long i,help=Ascii;
Cs    struct Window *wn;
Wm    struct RastPort *rp;
9z    short mox,moy,lochar=32,hichar=255;
Uh    BYTE proflag=1;
SC    void zahl_out();
sG3      wn=Make_Window(50,20,318,90,"SAVE FONT",flags,iflags,0);
XE       if(wn==0)return();
L5       rp=wn->RPort;
XO       SetAPen(rp,1);RectFill(rp,0,11,318,90);
Ih       Box(rp,2,12,315,88,0,JAM1);
8u       for(i=0;i<=9;i++)
tJ6         SetAPen(rp,0);
DH          RectFill(rp,sdat[i][0],sdat[i][1],sdat[i][2],sdat[i][3]);
DD          Box(rp,sdat[i][0],sdat[i][1],sdat[i][2],sdat[i][3],2,JAM1);
BV          Text_out(rp,sdat[i][0]+3,sdat[i][1]+3,3,0,JAM2,stxt[i],0);
uq3      s_info.Buffer=edrawer;
mA       s_info.UndoBuffer=NULL;
19       s_info.MaxChars=100;
U2       AddGadget(wn,&s_gad,-1);
bC       RefreshGadgets(&s_gad,wn,NULL);
ku       zahl_out(rp,lochar,hichar);
j7       Invert(rp,sdat[5]);
7o       FOREVER
zz6         while (ExaIntui(wn,&class,&code))
cM9            mox=wn->MouseX;moy=wn->MouseY;
CN             switch(class)
3FC               case CLOSEWINDOW:
71                ende:
ROF                  RemoveGadget(wn,&s_gad);
Ga                   CloseWindow(wn);
YO                   return();
AYC            case MOUSEBUTTONS:
SgF               if(code != SELECTDOWN)break;
I8                i=Prurec(mox,moy,sdat,2,9);
aQ                if(i<0)break;
Kl                SetAPen(rp,1);
nM                switch(i)
sAI                  case 2:
QpL                     YMatrix=(long)input_zahl(wn,61,36,2,"");
q2                      if(YMatrix<0 || YMatrix>50) YMatrix=8;
mv                      break;
yHI                  case 3:
JxL                     lochar=(short)input_zahl(wn,163,36,3,"");
3t                      if(lochar>hichar || lochar<0) lochar=32;
qz                      break;
4OI                  case 4:
8ZL                     hichar=(short)input_zahl(wn,273,36,3,"");
nu                      if(hichar<lochar || hichar>255) hichar=255;
u3                      break;
AVI                  case 5:
OaL                     if(proflag==1)break;
4N                      proflag=1;
Fp                      Invert(rp,sdat[5]);Invert(rp,sdat[6]);
z8                      break;
HdI                  case 6:
1aL                     if(proflag==0)break;
4M                      proflag=0;
Ku                      Invert(rp,sdat[5]);Invert(rp,sdat[6]);
4D                      break;
YyH                  case 7: case 8:
2GL                     Invert(rp,sdat[i]);
mV                      if(i==7)
8PO                        if(MakeFont(proflag,lochar,hichar)<0)
1FR                           Shure(2);
tWL                     Ascii=help;AsciiOut();
Yp                      Delay(5);
kd                      goto ende;
DM                      break;
bOH                  case 9:
BPI                     Invert(rp,sdat[i]);
1z                      GetFilename(edrawer,efname);
xv                      YMatrix=(long)atoi(efname);
ES                      Invert(rp,sdat[i]);
JS                      break;
U5C            RefreshGadgets(&s_gad,wn,NULL);
dn             zahl_out(rp,lochar,hichar);
100   /*---- Unterfunktion zu h0.7 ------*/
9o    void zahl_out(rp,lochar,hichar)
11    struct RastPort *rp;
Pu    short lochar,hichar;
JH    char htext[10];
dq3      sprintf(htext,"%2.0f",(float)YMatrix);
eE       Text_out(rp,61,36,1,0,JAM2,htext,0);
3A       sprintf(htext,"%3.0f",(float)lochar);
Zg       Text_out(rp,163,36,1,0,JAM2,htext,0);
Vm       sprintf(htext,"%3.0f",(float)hichar);
kt       Text_out(rp,273,36,1,0,JAM2,htext,0);
rLO   /*h0.8----------------- Font berechnen und schreiben -------------------------*/
40    BYTE MakeFont(pflag,lochar,hichar)
h9    BYTE pflag;
a5    short lochar,hichar;
dq    ULONG datei;
H3    BYTE mode,hmem[100],*nam;
OF    short modulo=0,anz=hichar-lochar+1,i,anz1,h1;
D1    USHORT *point=&ublock1[5];
yJ    void WriteLine(),WriteCharLoc(),WriteSpace(),WriteKern();
Nn3      mode=PruConf(edrawer,YMatrix);
JM       if(WriteFont(mode,edrawer,YMatrix)<0)return(-1);
Ip       sprintf(hmem,"%s/%1.0f",edrawer,(float)YMatrix);
tD       Shure(8);
RO       for(i=lochar;i<=hichar;i++)
Np6         Ascii=i;AsciiOut();
HE          fontbreit[i]=Breit(pflag,i);
```

```
2m         modulo+=fontbreit[i];
pT3     modulo/=8;modulo+=(4-modulo%4);
AY      h1=0x6e+modulo*YMatrix;
vz      anz1=modulo*YMatrix+anz*8+8;
6T      point++;
Wg      *point=(anz1+112)/4;point+=4;
DB      *point=(anz1+112)/4;point=&ublock1[27];
GF      *point=anz1;point++;
hk      *point=YMatrix;point++;
11      *point=98;
CZ      point++;
Bh      *point=(USHORT)XMatrix;point++;
xP      *point=(USHORT)BaseLine;point+=3;
mc      *point=256*lochar+hichar;point+=3;
nJ      *point=modulo;point+=2;
DY      *point=h1;point+=2;
11      *point=h1+anz*4+4;point+=2;
Ga      *point=h1+(anz*4+4)+(anz*2+2);
qm      point=drawmem;
Ga      Shure(8);
o1      datei=Open(hmem,MODE_NEWFILE);
SY      if(datei==0)return(-1);
Mh      Shure(9);
Gg      if(Write(datei,ublock1,142)!=142) goto abbruch;
Dc      /* ---- CharData schreiben -----*/
nd      for(i=0;i<=YMatrix-1;i++)
Ez6        BltClear(point,modulo,1);
wM         WriteLine(i,lochar,hichar);
9m         if(Write(datei,point,modulo)!=modulo) goto abbruch;
4L3     /* ---- CharLoc schreiben -----*/
J1      WriteCharLoc(lochar,hichar);
a0      if(Write(datei,point,anz*4+4)!=anz*4+4) goto abbruch;
VC      /* ---- CharSpace schreiben ----*/
P2      WriteSpace(lochar,hichar);
FX      if(Write(datei,point,anz*2+2)!=anz*2+2) goto abbruch;
OJ      /* ---- CharKern schreiben -----*/
Te      WriteKern(lochar,hichar);
Ia      if(Write(datei,point,anz*2+2)!=anz*2+2) goto abbruch;
eC      if(Write(datei,ublock2,46)!=46) goto abbruch;
VO      Close(datei);
ez      Shure(9);
JG      /* ---- Fast Direktory erzeugen -----*/
Rm      if(mode != 0)
h86        sprintf(hmem,"%s/fd.fdir",edrawer);
a8         DeleteFile(hmem);
UC3     if(mode==2)
TY6        nam=AllocMem(1024,MEMF_CLEAR);
Jw         if(nam != 0)
fD9           strcpy(hmem,edrawer);
3X            Parent(hmem);
TS            sprintf(hmem,"%s/fd.fdir",hmem);
HJ            datei=Open(hmem,MODE_OLDFILE);
I5            if(datei != 0)
jcC              Close(datei);
60               Shure(10);
9d               Parent(hmem);
Lt               GetDir(nam,hmem,1);
9R               Shure(10);
Rx9           FreeMem(nam,1024);
ID3     return(0);
YEO  abbruch:
zK3     Shure(9);
s1      Close(datei);
wX      return(-1);
C10  /*--- Unterfunktionen zu h0.8-----*/
2x   void WriteLine(i,lochar,hichar)
r1   short i,lochar,hichar;
0z   register short j,k;
oJ   register ULONG *adr;
eN   register long count=0;
y3   long *point=drawmem;
tQ3     for(j=lochar;j<=hichar;j++)
G16        adr=cmatrix+100*j+2*i;
fk         for(k=0;k<=fontbreit[j]-1;k++,count++)
9I9           if(TestBit(adr,k))
fJC              SetBit(point,count);
vi9           else
OrC              ClrBit(point,count);
MA0  /*--------------------------------*/
ZM   void WriteCharLoc(lochar,hichar)
5a   short lochar,hichar;
fp   register short *point=drawmem,offs=0,i;
oN3     for(i=lochar;i<=hichar;i++)
kC6        Ascii=i;AsciiOut();
x0         *point=offs;point++;
Jc         *point=fontbreit[i];point++;
Q6         offs+=fontbreit[i];
jw3     *point=0;point++;*point=0;
Fq0  /*-----------------------------*/
VM   void WriteKern(lochar,hichar)
Fk   short lochar,hichar;
YG   register short *point=drawmem,i;
yX3     for(i=lochar;i<=hichar;i++)
uM6        Ascii=i;AsciiOut();
Vu         *point=1;
Zw         point++;
Tr3     *point=0;
Zx0  /*----------------------------*/
NV   void WriteSpace(lochar,hichar)
Ot   short lochar,hichar;
hP   register short *point=drawmem,i;
7g3     for(i=lochar;i<=hichar;i++)
3V6        Ascii=i;AsciiOut();
KP         if(fontbreit[i]==0)
1g9           *point=XMatrix;
0B6        else
nT9           *point=fontbreit[i]+1;
186        point++;
f33     *point=0;
0X0  /*h0.9--------------- Ermittelt die Breite eines Zeichens
     ----------------*/
Bn   short Breit(pflag,ascii)
iA   BYTE pflag;
JM   short ascii;
q4   ULONG *adr=cmatrix+100*ascii;
8Z   long i,j,flag=0;
zq3     if(pflag==0)return(XMatrix);
gZ      for(i=XMatrix-1;i>=0;i--)
h96        for(j=0;j<=YMatrix;j++)
1C9           if(TestBit(adr,j*64+i))
JhC              flag=1;
ob3     if(flag==1)break;
35      if(flag==0)return(0);
ms      return(i+1);
tf0  /*h0.10--------------- Prüft ob Diskfont vorhanden -------
     ----------------*/
1M   BYTE PruConf(drawer,high)
0j   char drawer[];
fc   short high;
s5   ULONG datei;
s2   char hmem[100],h;
oR   struct FileLock *lock;
d73     sprintf(hmem,"%s/%1.0f",drawer,(float)high);
Zb      datei=Open(hmem,MODE_OLDFILE);
04      if( datei != 0) { Close(datei);return(0); }
n0      lock=Lock(drawer,ACCESS_READ);
ZV      if(lock != 0) h=1;
sf      else
Nv6        lock=CreateDir(drawer);
9V         h=2;
RW3     UnLock(lock);
C1      return(h);
Dg0  /*h0.11----------------Erzeugt den File xxxx.font --------
     ----------------*/
jr   BYTE WriteFont(mode,drawer,high)
E7   BYTE mode;
I1   char drawer[];
xu   short high;
mW   char puffer[264],filename[100];
tq   long help;
Wo   short i;
DQ   ULONG datei;
MK3     if(mode==0)return(0);
5G      if (datei==0) {Shure(11);
rxI                    return(-1);}
RC3     for(i=0;i<=263;i++) puffer[i]=0;
xJ      puffer[0]=0x0f;puffer[3]=1;puffer[263]=0x62;puffer[261]
        =high;
Lf      for(i=strlen(drawer);i>=0;i--)
Ms6        if(drawer[i]=='/' || drawer[i]==':')
JS9           break;
xz3     sprintf(filename,"%s.font",drawer);
5i      sprintf(&puffer[4],"%s/%1.0f",&drawer[i+1],(float)high)
        ;
```

Listing 2. (Fortsetzung)

```
   switch(mode)
      case 1:
         datei=Open(filename,MODE_OLDFILE);
         if(datei==0)return(-1);
         if(Read(datei,&help,4)!=4) goto abbruch;
         help++;
         Seek(datei,0,OFFSET_BEGINNING);
         if(Write(datei,&help,4)!=4) goto abbruch;
         Seek(datei,0,OFFSET_END);
         if(Write(datei,&puffer[4],260)!=260)goto abbruch;
         break;
      case 2:
         datei=Open(filename,MODE_NEWFILE);
         if(datei==0) {shure(11);
                       return(-1);}
         if(Write(datei,puffer,264)!=264) goto abbruch;
         break;
   Shure(11);
   Close(datei);
   return(0);
abbruch:
   Shure(11);
   Close(datei);
   return(-1);
/*h0.12--------- Ermittelt übergeordnetes Verzeichnis ----
--------------*/
void Parent(name)
char *name;
register short i,flag;
   for(i=strlen(name);i>=0;i--)
      if(name[i]=='/') { flag=0;break; }
      if(name[i]==':') { flag=1;break; }
   if(flag==1)i++;
   name[i]=0;
/*h0.13----------- Zeichnet kleines Image ----------------
----*/
void DrawLittle(mode,x,y,adr)
BYTE mode;
short x,y;
register ULONG *adr;
register ULONG *adr1=drawmem;
register short i;
   for(i=0;i<=99;*adr1=*adr,adr++,adr1++,i++);
   oima->ImageData=mima->ImageData=drawmem;
   if(mode==0)
      DrawImage(rast,oima,x,y);
   else
      DrawImage(rast,mima,x,y);
/*--------------------------------------------------------
---------------*/
```

(C) 1988 M&T

Listing 2. (Schluß)

Programmname:	h1.c
Computer:	siehe Listing 1
Sprache:	C

```
Programm : h1.c

---------------

/*---(FontDesigner)------ File h1.c ---------*/
#include <exec/memory.h>
#include <intuition/intuition.h>
#include <intuition/intuitionbase.h>
#include <libraries/dos.h>
#include <graphics/gfxmacros.h>
#include "header.h"
/*-------------------- Externe Variablen aus main ---------
--------------*/
extern struct IntuitionBase *IntuitionBase;
extern struct GfxBase *GfxBase;
extern ULONG DiskfontBase,DosBase;
extern BYTE *drawmem;
extern long *cmatrix,Ascii;
/*---------------------- Daten für input_zahl() ----------
--------------*/
unsigned char bmaske[]=
{
   128,64,32,16,8,4,2,1
};
UBYTE rawkey[]={1,2,3,4,5,6,7,8,9,10,57,18,58,68,61,62,63,4
5,46,47,29,30,31,
          15,60,74,67,70,65,69,79,78};
UBYTE ascii[]={49,50,51,52,53,54,55,56,57,48,46,101,45,13,5
5,56,57,52,53,54,
          49,50,51,48,46,45,13,1,8,27,2,3};
/*h1.1--------------------- Zeichnet eine Linie ----------
---------------*/
void Line(rp,xa,ya,xe,ye,co,drmd)
struct RastPort *rp;
short xa,ya,xe,ye;
BYTE co,drmd;
{
   SetAPen(rp,co);SetDrMd(rp,drmd);
   Move(rp,xa,ya);
   Draw(rp,xe,ye);
}
/*h1.2--------------------- Zeichnet ein Rechteck ---------
--------------*/
void Box(rp,xa,ya,xe,ye,co,drmd)
struct RastPort *rp;
short xa,ya,xe,ye;
BYTE co,drmd;
{
   SetAPen(rp,co);SetDrMd(rp,drmd);
   Move(rp,xa,ya);
   Draw(rp,xe,ya);Draw(rp,xe,ye);
   Draw(rp,xa,ye);Draw(rp,xa,ya);
}
/*h1.3--------------------- Installiert ein Fenster -------
--------------*/
ULONG Make_Window (x,y,w,h,name,flags,iflags,screen)
short x,y,w,h;
char *name;
ULONG flags,iflags;
struct Screen *screen;
{
struct Window *wn;
struct NewWindow NewWindow;
   NewWindow.LeftEdge=x;
   NewWindow.TopEdge=y;
   NewWindow.Width=w;
   NewWindow.Height=h;
   NewWindow.DetailPen=0;
   NewWindow.BlockPen=1;
   NewWindow.Title=name;
   NewWindow.Flags=flags;
   NewWindow.IDCMPFlags=iflags;
   if(screen==NULL)
      NewWindow.Type=WBENCHSCREEN;
   else NewWindow.Type=CUSTOMSCREEN;
   NewWindow.FirstGadget=NULL;
   NewWindow.CheckMark=NULL;
   if (screen==NULL)
      NewWindow.Screen=IntuitionBase->FirstScreen;
   else NewWindow.Screen=screen;
   NewWindow.BitMap=NULL;
   NewWindow.MinWidth=w;
   NewWindow.MinHeight=h;
   NewWindow.MaxWidth=w;
   NewWindow.MaxHeight=h;
      wn=OpenWindow(&NewWindow);
      return(wn);
}
/*h1.4------------------- Holt Intuition-Message ----------
--------------*/
BYTE ExaIntui (wn,class,code)
struct Window *wn;
ULONG *class;
USHORT *code;
{
struct IntuiMessage *mesg;
   mesg=GetMsg(wn->UserPort);
   if (mesg != NULL)
   {
      *class=mesg->Class;
      *code=mesg->Code;
      ReplyMsg(mesg);
      return(1);
   }
   return(0);
}
```

```
RJ  /*h1.5---------------------- Gibt Text aus ----------------------------*/
xF  void Text_out (rp,x,y,apen,bpen,drmd,text,maxlen)
r7  struct RastPort *rp;
t1  short x,y,maxlen;
nb  BYTE apen,bpen,drmd;
0g  char *text;
Z2  {
193     Move(rp,x,y+rp->TxBaseline);
2p      SetAPen(rp,apen);SetBPen(rp,bpen);SetDrMd(rp,drmd);
9n      if(maxlen==0)Text(rp,text,strlen(text));
gH      else Text(rp,text,maxlen);
iD0 }
pE  /*h1.6------------- Zeichnet ein ausgefülltes Polygon ------------------*/
uQ  BYTE FDrawPolygon(rp,koord,co,drmd)
3J  struct RastPort *rp;
ya  short koord[];
gi  BYTE co,drmd;
kD  {
V0  struct AreaInfo ainfo;
Kq  ULONG speicher;
Z4  short abuffer[250],i;
bg  struct TmpRas tmp;
Dj3     speicher=drawmem;
2V      InitArea(&ainfo,abuffer,50);
zm      rp->TmpRas=(struct TmpRas *)InitTmpRas(&tmp,speicher,20480);
IS      rp->AreaInfo=&ainfo;
zm      SetDrMd(rp,drmd);SetAPen(rp,co);
y5      AreaMove(rp,koord[0],koord[1]);
5S      i=2;
yE      while(koord[i]>=0)
xQ      {
dx6        AreaDraw(rp,koord[i],koord[i+1]);
sF         i+=2;
4Z3     }
fW      AreaEnd(rp);
Rv      rp->TmpRas=NULL;rp->AreaInfo=NULL;
ea      return(1);
8d0 }
4p  /*h1.7------------------ Ermittelt Mausklickfeld -------------------------*/
yh  BYTE Prurec (mox,moy,feld,anf,end)
FV  register short mox,moy;
nG  short feld[][4];
e0  register BYTE anf,end;
Ad  {
27  register BYTE i;
wF3     for (i=anf;i<=end;i++)
Dg      {
yf6        if (mox >= feld[i][0] && mox <= feld[i][2] && moy >= feld[i][1]
CHC             && moy <= feld[i][3])
TJ9          return(i);
Lq3     }
P0      return(-1);
Ns0 }
ak  /*h1.8------------------ Eingabe einer double Zahl ----------------------*/
Ja  double input_zahl(wind,xa,ya,maxlen,defs)
6N  struct Window  *wind;
sx  short xa,ya;
N1  UBYTE maxlen;
Xj  char *defs;
Qt  {
8s  struct IntuiMessage *msg;
o4  struct RastPort *rp;
Ln  ULONG class,iflags;
hq  UBYTE code,i,pruef(),cpos=0;
sh  char ps[31];
0A  double atof();
G7  void crs_out();
wA  iflags=wind->IDCMPFlags;
aD  ModifyIDCMP(wind,MOUSEBUTTONS|RAWKEY);
vW  maxlen--;
5f  rp=wind->RPort;
D3  for (i=0;i<=29;i++)ps[i]=32;
ll  ps[30]=0;
qx  ya+=rp->TxBaseline;
WU  Move(rp,xa,ya);SetDrMd(rp,JAM2);Text(rp,ps,maxlen+1);
MN  if(strlen(defs) !=0)
hA  {
ib3     strcpy(ps,defs);
ke      cpos=strlen(defs);
vz      ps[cpos]=32;
pK0 }
LW  ps[maxlen+1]=0;
HX  Move(rp,xa,ya);SetDrMd(rp,JAM2);Text(rp,ps,strlen(ps));
AR  crs_out (rp,cpos,xa,ya);
R83     FOREVER
qJ      {
4J9           while(msg=(struct IntuiMessage *)GetMsg(wind->UserPort))
sL            {
XyC              class=msg->Class;
GG               code=msg->Code;
rz               ReplyMsg(msg);
ew               switch (class)
xQ               {
UsF                 case MOUSEBUTTONS:
m0L                       if(code != SELECTDOWN)break;
qk                        crs_out(rp,cpos,xa,ya);
Sd                        ModifyIDCMP(wind,iflags);
6B                        return (atof(ps));
FhF                 case RAWKEY:
ba                  code=pruef(code);
NM                  if (code==128)break;
1b                  switch (code)
7a                  {
iiI                    case 1: /* Delete */
VML                       if(cpos==maxlen)
Ad                        {
0Y0                          ps[cpos]=32;ps[cpos+1]=0;break;
G1L                       }
k8                        for(i=cpos;i<=(maxlen-1);i++)ps[i]=ps[i+1];
Y1                        ps[maxlen]=32;
EN                        break;
X4I                    case 8: /* Backspace */
Q0L                       if(cpos==0)break;
Ax                        cpos--;
qE                        for(i=cpos;i<=(maxlen-1);i++)ps[i]=ps[i+1];
er                        ps[maxlen]=32;
KT                        break;
OEI                    case 2: /* Cursor left */
JEL                       if(cpos>0)cpos--;
NW                        break;
PgI                    case 3: /* Cursor right */
REL                       if(cpos<maxlen)cpos++;
QZ                        break;
xpI                    case 13: /* Carrige Return */
JDL                       crs_out(rp,cpos,xa,ya);
v6                        ModifyIDCMP(wind,iflags);
Ze                        return (atof(ps));
IiI                    case 27: /* ESC */
PzL                       for(i=0;i<=maxlen;i++)ps[i]=32;
4s                        cpos=0;
Yh                        break;
MGI                    default:
1KL                       for(i=maxlen;i>=(cpos+1);i--)ps[i]=ps[i-1];
1B                        ps[cpos]=code;
CN                        ps[maxlen+1]=0;
9e                        if (cpos<maxlen)cpos++;
JEI                 }
ou                  Move(rp,xa,ya);
8N                  SetDrMd(rp,JAM2);
3T                  Text(rp,ps,strlen(ps));
ZT                  crs_out(rp,cpos,xa,ya);
oJC              }
pK9           }
qL6     }
rM0 }
rj  /*----- Unterfunktionen zu h1.8 ---------*/
i7  void crs_out(rp,cpos,xa,ya)
CS  struct RastPort *rp;
Pu  UBYTE cpos;
NS  short xa,ya;
tM  {
lq  register BYTE i;
cJ3     Move(rp,xa+TextLength(rp," ",1)*cpos,ya);
```

Listing 3. Geben Sie »h1.c« bitte mit dem Checksummer ein

```
4Q      SetDrMd(rp,COMPLEMENT|INVERSVID);
nk      Text(rp," ",1);
2X0  }
8v   UBYTE pruef(code)
w9   USHORT code;
1U   {
uC   register UBYTE i;
ur3     for(i=0;i<=31;i++)
4X      {
676        if(code==rawkey[i]) return(ascii[i]);
Af3     }
kb      return(128);
Ch0  }
S4   /*h1.9--------------------- Öffnet Libraries --------------------------*/
dL   BYTE Oplibs()
Be   {
1n3     IntuitionBase=OpenLibrary("intuition.library",0L);
XH      GfxBase=OpenLibrary("graphics.library",0L);
4Q      DosBase=OpenLibrary("dos.library",0L);
72      DiskfontBase=OpenLibrary("diskfont.library",0L);
J1      if(DiskfontBase==0)
Hk      {
FV6        Shure(4);
eD         CloseLibrary(IntuitionBase);
oK         CloseLibrary(GfxBase);
qE         exit(0);
Qv3     }
up      return(0);
Sx0  }
hL   /*h1.10--------------------- Schließt Libraries ------------------------*/
u0   void Cllibs()
Ru   {
PD3     CloseLibrary(DosBase);
ck      CloseLibrary(DiskfontBase);
yU      CloseLibrary(GfxBase);
qP      CloseLibrary(IntuitionBase);
a50  }
es   /*h1.11------------------- Invertier BS_Bereich -----------------------*/
24   void Invert(rp,dat)
vB   struct RastPort *rp;
tg   short dat[];
b4   {
KF3     SetDrMd(rp,COMPLEMENT);
02      RectFill(rp,dat[0]+1,dat[1]+1,dat[2]-1,dat[3]-1);
1D0  }
V1   /*h1.12--------------------- Prüft ob Bit gesetzt -----------------------*/
N7   BYTE TestBit(adr,bioff)
zY   register UBYTE *adr;
Cs   register long bioff;
jC   {
Bn3     adr+=bioff>>3;
AX      if((*adr & bmaske[bioff%8]) != 0)
NJ6        return(1);
KF3     return(0);
sN0  }
qm   /*h1.13--------------------- Bestimmtes Bit setzen ---------------------*/
kc   void SetBit(adr,bioff)
91   register UBYTE *adr;
M2   register long bioff;
tM   {
Lx3     adr+=bioff>>3;
8P      *adr|=bmaske[bioff%8];
0V0  }
gY   /*h1.14-------------------- Bestimmtes Bit löschen ---------------------*/
ZG   void ClrBit(adr,bioff)
Hq   register UBYTE *adr;
UA   register long bioff;
1U   {
T53     adr+=bioff>>3;
bU      *adr&=**ßbmaske[bioff%8];
8d0  }
4M   /*h1.15-------------------- Zeichensatz verschieben --------------------*/
19   short mdat[][4]=
7a   {
oh3     { 10,16,128,28 },{ 138,16,262,28 },{ 86,33,188,45 },
ea      { 10,50,64,62 },{ 208,50,262,62 },
XT0  };
7t   char *mtxt[]=
Cf   {
NZ3     "FROM ASCII:","UNTIL ASCII:","TO ASCII:"," OK","CANCEL"
        ,
bX0  };
Y5   short moveas[3]={ 65,128,128 };
Hz   void MoveAscii()
Hk   {
L1   struct Window *wn;
fv   struct RastPort *rp;
XN   ULONG flags=WINDOWDRAG|ACTIVATE;
AU   ULONG class;
IV   USHORT code;
xD   short i,mox,moy,*point,offs;
xC   void moveout(),MAusf();
9c3     wn=Make_Window(120,100,280,71,"MOVE",flags,MOUSEBUTTONS,
        0);
iP      if(wn==0)return();
WG      rp=wn->RPort;
SG      SetAPen(rp,1);RectFill(rp,0,11,280,71);
30      Box(rp,2,12,277,69,0,JAM1);
41      for(i=0;i<=4;i++)
Vy      {
5V6        SetAPen(rp,0);
lv         RectFill(rp,mdat[i][0],mdat[i][1],mdat[i][2],mdat[i][
           3]);
lr         Box(rp,mdat[i][0],mdat[i][1],mdat[i][2],mdat[i][3],2,
           JAM1);
Ne         Box(rp,mdat[i][0]-1,mdat[i][1],mdat[i][2]+1,mdat[i][3
           ],2,JAM1);
QS         Text_out(rp,mdat[i][0]+3,mdat[i][1]+3,3,0,JAM2,mtxt[i
           ],0);
fA3     }
1X      moveout(rp);
Fw      FOREVER
e7      {
ke6        while(ExaIntui(wn,&class,&code))
g9         {
zS9           if(code!=SELECTDOWN)break;
XD            i=Prurec(wn->MouseX,wn->MouseY,mdat,0,4);
cS            if(i<0)break;

oN            switch(i)
1E            {
niC              case 0: case 1: case 2:
6NF                 point=&moveas[i];
ce                  offs=mdat[i][0]+strlen(mtxt[i])*8+3;
U2                  *point=(short)input_zahl(wn,offs,mdat[i][1]+
                    3,3,"");
JQ                  if(*point<0)*point=0;
AT                  if(*point>255)*point=255;
r0                  break;
3MC              case 3:
d1F                 Invert(rp,mdat[i]);
Qa                  MAusf();
8SC              case 4:
goF                 Invert(rp,mdat[i]);
Kb                  Delay(5);
dx                  CloseWindow(wn);
v1                  return();
5a9           }
Rx6        moveout(rp);
7c         }
8d3     }
9e0  }
yT   /*---- Unterfunktion zu h1.15 ----*/
01   void moveout(rp)
Uk   struct RastPort *rp;
9c   {
uZ   short offs,i;
H5   char text[5];
gL3     for(i=0;i<=2;i++)
Dg      {
Yj6        sprintf(text,"%3.0f",(float)moveas[i]);
35         offs=mdat[i][0]+strlen(mtxt[i])*8+3;
oB         Text_out(rp,offs,mdat[i][1]+3,3,0,JAM2,text,0);
Lq3     }
Mr0  }
Bg   /*---- Unterfunktion zu h1.15 ----*/
sX   void MAusf()
Lo   {
ZD   short mofa=moveas[0],moua=moveas[1],mota=moveas[2],help;
```

```
FA  register short j,i,step=1;
6p  long *adr,*adr1;
Kn  long merker=Ascii;
Yf3    if(mofa==moua)return();
Zp     if(mofa>moua) { help=mofa;mofa=moua;moua=help; }
K2     if(mota > mofa && mota < moua)
Tw     {
aJ6       mota+=(moua-mofa);
YA        help=mofa;mofa=moua;moua=help-1;
1A        step=-1;
b63    }
Cp     else moua++;
xC     for(i=mofa;i!=moua;i+=step,mota+=step)
a3     {
5R6       if(mota>255)continue;
oY        adr=cmatrix+100*i;
Sm        adr1=cmatrix+100*mota;
Ww        for(j=0;j<=99;j++,adr1++)*adr1=adr[j];
In        Ascii=i;
XQ        AsciiOut();
1G3    }
3e     Ascii=merker;
aT     AsciiOut();
oJ0 }
bP  /*------------------------------------------------------------
    --------------*/
(C) 1987 M&T
```

Listing 3. (Schluß)

Programmname:	h2.c
Computer:	siehe Listing 1
Sprache:	C

```
Programm : h2.c

---------------

RM0 /*--(FontDesigner)--------- File h2.c ------------*/
9E  #include <exec/memory.h>
Rg  #include <intuition/intuition.h>
L9  #include <intuition/intuitionbase.h>
3q  #include <libraries/dos.h>
i5  #include <graphics/gfxmacros.h>
MB  #include "header.h"
OB  /*-------------------- Externe Variablen aus main ---------
    ----------*/
Ip  extern long XMatrix,YMatrix,BaseLine,HelpLine,Ascii,DMode;
9R  extern long Quadrant;
OU  extern struct TextFont *tftopaz,*tfuser;
FJ  extern struct Window *wind;
ek  extern struct RastPort *rast;
Vi  extern ULONG *cmatrix,maske[],cmemo[],scmemo[];
5d  extern UBYTE bmaske[],*drawmem;
nB  extern struct Image *oima,*drima,*mima;
EJ  /*---------------------- Daten für Hauptdisplay ----------
    -----------*/
ls  short recdat[][4]=
Fi  {
oH3    { 0,11,512,211 },
Ua     { 456,216,505,228}, /* Change Ascii */
Eh     { 456,228,505,239},
i8     { 456,239,505,250 },
m6     { 520,187,566,199 }, /*  4 */
2y     { 566,187,597,199 },
9z     { 597,187,628,199 },
us     { 520,204,566,216 }, /*  7 */
4b     { 566,204,597,216 },
Bc     { 597,204,628,216 },
Bu     { 520,221,566,233 }, /* 10 */
vQ     { 566,221,597,233 },
2R     { 597,221,628,233 },
sl     { 520,238,566,250 }, /* 13 */
W8     { 566,238,597,250 },
d9     { 597,238,628,250 },
Wa     { 594,13,628,27 }, /* 16 */
CF     { 594,35,628,49 }, /* 17 */
zg     { 594,57,628,71 }, /* 18 */
fL     { 594,79,628,93 }, /* 19 */
7d     { 594,101,628,115 }, /* 20 */
lr     { 520,145,556,163 },  /* 21 */
72     { 592,145,628,163 },
Bm     { 556,127,592,145 },
gF     { 556,163,592,181 }, /* 24 */
Cn     { 556,145,574,154 },
9u     { 574,145,592,154 },
Cn     { 556,154,574,163 },
tg     { 574,154,592,163 }, /* 28 */
5J     { 10,216,64,228 },  /* MIRR_B */
HE     { 10,238,64,250 },  /* MIRR_H */
JC     { 79,216,133,228 }, /* ROTATE */
Bc     { 302,216,356,228}, /* UNDO */
2s     { 79,238,133,250 }, /* INVERS */
UB     { 233,216,287,228 },      /* LOAD */
Mr     { 148,216,218,228 },/* EXPAND_X */
Pu     { 148,238,218,250 },/* EXPAND_Y */
9T     { 233,238,287,250},  /* SAVE */
UA     { 371,216,441,228 }, /* PRG */
bK     { 371,238,441,250 }, /* SHOW */
JG     { 302,238,356,250 }, /* MOVE */
IE0 };
As  char *rectxt[]=
xQ  {
kS3    "","","","",
r0     "XM=","","","YM=","","","BL=","","","HL=","","",
ap     "ALL","MAT","DEL","SNP","CPY",
nV     "","","","",
oW     "","","","",
1o     "MIRR_H","MIRR_B","ROTATE","UNDO","INVERS","LOAD","EXPAN
    D_X","EXPAND_Y",
ZX     "SAVE","PROGRAMM","SHOW","MOVE",
S00 };
4s  char txtoff[][2]=
7a  {
te3    {0,0},{11,3},{0,0},{0,0},
PM     {3,3},{0,0},{0,0},
QN     {3,3},{0,0},{0,0},
RO     {3,3},{0,0},{0,0},
SP     {3,3},{0,0},{0,0},
HB     {5,4},{5,4},{5,4},{5,4},{5,4},  /* 16-20 */
yz     {0,0},{0,0},{0,0},{0,0},        /* 21-24 */
ZO     {0,0},{0,0},{0,0},{0,0},        /* 25-28 */
L7     {3,3},{3,3},{3,3},{11,3},
9x     {3,3},{11,3},{3,3},{3,3},
Yu     {11,3},{3,3},{19,3},{11,3},
gc0 };
0a  short pfeil[][7]=
Lo  {
zn3    { 528,154,545,146,545,162,-2 },
d8     { 620,154,603,146,603,162,-2 },
I4     { 574,129,560,143,588,143,-2 },
9y     { 574,179,560,165,588,165,-2 },
Ap     { 576,193,586,188,586,198,-2 },
us     { 617,193,607,188,607,198,-2 },
4s     { 581,206,573,214,589,214,-2 },
JP     { 612,214,604,206,620,206,-2 },
mm     { 581,223,573,231,589,231,-2 },
RJ     { 612,231,604,223,620,223,-2 },
oA     { 576,244,586,239,586,249,-2 },
YD     { 617,244,607,239,607,249,-2 },
pc     { 480,229,472,237,488,237,-2 },
DM     { 480,249,472,241,488,241,-2 },
xt0 };
ZI  char wtitle[]="F O N T  DESIGNER    Progammed by Richard Ar
    etz";
aZ  /*h2.1------------------ Erzeugt das Hauptdisplay ---------
    ------------*/
OX  BYTE Display()
e7  {
3h  ULONG flags=WINDOWCLOSE| ACTIVATE| REPORTMOUSE| SMART_REFRESH|
NrC             RMBTRAP| WINDOWDEPTH;
KI0 ULONG iflags=CLOSEWINDOW| MOUSEBUTTONS| VANILLAKEY;
3L  short i;
0G3    wind=Make_Window(0,0,636,256,wtitle,flags,iflags,0);
```

Listing 4. »h2.c« bitte mit dem Checksummer eingeben. Beachten Sie die Anweisungen im Text und in Tabelle 2a beziehungsweise 2b (Seite 54).

```
MO       if(wind==0) return(-1);
Tb       rast=wind->RPort;
Sv       SetAPen(rast,3);RectFill(rast,0,11,640,256);
Xt       SetAPen(rast,0);
U9       RectFill(rast,0,11,512,211);
b5       RectFill(rast,521,14,587,65);
5t       RectFill(rast,521,70,587,121);
AX       for(i=21;i<=24;i++)
sL       {
rk6         RectFill(rast,recdat[i][0]+4,recdat[i][1]+2,
QCN                        recdat[i][2]+4,recdat[i][3]+2);
zU3      }
pW       Box(rast,0,11,512,211,2,JAM1);
qo       for(i=1;i<=40;i++)
yR       {
Jj6         if ( !(i>20 && i<29) )
0T          {
179            SetAPen(rast,0);
0t             RectFill(rast,recdat[i][0]+4,recdat[i][1]+2,
ZLN                          recdat[i][2]+4,recdat[i][3]+2);
8d6         }
uH          SetAPen(rast,1);
4k          RectFill(rast,recdat[i][0],recdat[i][1],recdat[i][2],
            recdat[i][3]);
4g          Box(rast,recdat[i][0],recdat[i][1],recdat[i][2],recda
            t[i][3],2,JAM1);
rW          Text_out(rast,recdat[i][0]+txtoff[i][0],recdat[i][1]+
            txtoff[i][1],0,1,
KVF                   JAM2,rectxt[i],0);
Ej3      }
IZ       for (i=0;i<=13;i++)
t46         FDrawPolygon(rast,pfeil[i],2,JAM1);
b03      AsciiOut();Matout();Mem_to_Matrix();
fh       Invert(rast,recdat[17-DMode]);
RN       Invert(rast,recdat[25+Quadrant]);
ni       return(0);
Lq0 }
G6  /*h2.2---------------- Aktuellen ascii-code ausgeben ------
    -----------*/
5T  void AsciiOut()
Kn  {
RF  char text[5];
Ui3    sprintf(text,"%3.0f",(float)Ascii);
T5     Text_out(rast,recdat[1][0]+txtoff[1][0],recdat[1][1]+txt
       off[1][1],
CMC             0,1,JAM2,text,0);
Ty0 }
ZT  /*h2.3------------------ Werte für Matrix ausgeben --------
    ----------*/
a4  void Matout()
Sv  {
Mq  char text[10],i;
DA  long help;
pR3    for(i=4;i<=13;i+=3)
Wz     {
bA6       switch(i)
Y1        {
6j9          case 4: help=XMatrix;break;
Iz           case 7: help=YMatrix;break;
ye           case 10: help=BaseLine;break;
Dg           case 13: help=HelpLine;break;
hC6       }
J4        sprintf(text,"%s%2.0f",rectxt[i],(float)help);
J1        Text_out(rast,recdat[i][0]+txtoff[i][0],recdat[i][1]+
          txtoff[i][1],
UeF                0,1,JAM2,text,0);
1G3    }
mH0 }
cI  /*h2.4--------------- Aktuellen Ascii-code verändern ------
    -------------*/
Xy  void ChangeAscii(mode)
tm  BYTE mode;
mF  {
cw  ULONG class;
kx  USHORT code;
cB  register short i;
bR3    if(mode==0)
rK     {
cy6       SetAPen(rast,0);
G0        Ascii=(long)input_zahl(wind,recdat[1][0]+txtoff[1][0]
          ,
rxT                                recdat[1][1]+txtoff[1][1],3,""
                                   );
oy6       if(Ascii<0 || Ascii>255)Ascii=65;
ng        AsciiOut();
6K        Mem_to_Matrix();
tj        return();
3Y3    }
RL     Invert(rast,recdat[mode+1]);
n5     Delay(10);
eL     FOREVER
3W     {
nX6       switch (mode)
5Y        {
CT9          case 1:
jPC             if(Ascii<255)Ascii++;
pC              else Ascii=0;
8H              break;
Ia9          case 2:
HCC             if(Ascii>0)Ascii--;
uo              else Ascii=255;
CL              break;
In6       }
6z        AsciiOut();
uk        DrawLittle(0,0,0,cmatrix+100*Ascii);
Xm        Delay(3);
z4        while(ExaIntui(wind,&class,&code))
Jm        {
y99          switch(class)
Lo           {
sGC             case MOUSEBUTTONS:
WkF                Mem_to_Matrix();
pj                 Invert(rast,recdat[mode+1]);
KA                 return();
Uz9          }
V06       }
W13    }
X20 }
L4  /*h2.5-------------------- Matrixdaten verändern ----------
    -------------*/
4n  void ChangeMatrix(var)
AG  BYTE var;
X0  {
Nh  ULONG class;
Vi  USHORT code;
ZF  long *point,pmax,mode=(var-1)%3;
Cb3    Mabox(0);
ga     switch (var)
d6     {
JQ6       case 4: case 5: case 6:
aW9          point=&XMatrix;pmax=64;break;
p56       case 7: case 8: case 9:
SK9          point=&YMatrix;pmax=50;break;
AL6       case 10: case 11: case 12:
ud9          point=&BaseLine;pmax=50;break;
jo6       case 13: case 14: case 15:
dE9          point=&HelpLine;pmax=64;break;
qL3    }
Y0     if(mode==0)
oH     {
Zv6       SetAPen(rast,0);
NT        *point=(long)input_zahl(wind,recdat[var][0]+26,recdat
          [var][1]+3,2,"");
SW        if(*point<1)*point=1;
eW        if(*point>pmax)*point=pmax;
QP        Matout();
3H        Mem_to_Matrix();
qg        return();
0V3    }
da     Invert(rast,recdat[var]);
k2     Delay(10);
bI     FOREVER
0T     {
c16       Mabox(0);
1V        switch (mode)
3W        {
AR9          case 1:
hrC             if(*point>0)*point-=1;
dJ              else *point=pmax;
6F              break;
GY9          case 2:
uQC             if(*point<pmax)*point+=1;
AM              else *point=1;
Fk6       }
qG        Mabox(1);
kj        Matout();
```

```
Rf         Delay(2);
Va         while (ExaIntui(wind,&class,&code))
Gj         {
zH9           switch (class)
Il            {
pDC              case MOUSEBUTTONS:
OxF                 Invert(rast,recdat[var]);
Ui                  Mem_to_Matrix();
H7                  return();
Rw9           }
Sx6        }
Ty3     }
Uz0 }
7Z  /*h2.6------------------ Zeichnet die Matrixbox -----------
    ------------*/
a0  void Mabox(mode)
bU  BYTE mode;
Ux  {
f3  short xe=XMatrix*8,ye=11+YMatrix*4,yb=11+BaseLine*4,xh=Help
    Line*8;
on3    Box(rast,0,11,xe,ye,mode+2,JAM1);
p5     Line(rast,0,yb,xe,yb,mode+2,JAM1);
M0     Line(rast,xh,11,xh,ye,mode+2,JAM1);
d80 }
H6  /*h2.7-------------Setzt einen Punkt in die Matrix --------
    -----------*/
iR  void SetMPoint(sp,ze,mode)
Nt  BYTE sp,ze,mode;
d6  {
gb  short xa=sp*8+1,ya=11+ze*4+1,spmax=63,zemax=49;
3V  ULONG *adr=cmatrix+100*Ascii+2*ze;
qw3    if(DMode==0) { spmax=XMatrix-1;zemax=YMatrix-1; }
fQ     if(sp>spmax || sp<0)return();
st     if(ze<0 || ze>zemax)return();
zI     SetAPen(rast,mode);SetDrMd(rast,JAM1);
R3     RectFill(rast,xa,ya,xa+6,ya+2);
xX     WritePixel(rast,522+sp,15+ze);
ga     if(sp>=32){ adr++;sp-=32; }
eQ     if(mode==1)
ir6       *adr|=maske[sp];
iV3    else
xz6       *adr&=**ßmaske[sp];
vQ0 }
7P  /*h2.8------------- Haupfunktion in Matrix zeichnen -------
    -------------*/
qA  void DrawInto(mode)
2v  BYTE mode;
v0  {
l5  ULONG class;
t6  USHORT code;
LZ  short mox,moy;
x6  ULONG iflags=wind->IDCMPFlags;
nm  void SetMPoint();
pq3    ModifyIDCMP(wind,wind->IDCMPFlags|=MOUSEMOVE);
DD     SetMPoint(wind->MouseX/8,(wind->MouseY-11)/4,mode);
fM     FOREVER
4X     {
LQ6       while (ExaIntui(wind,&class,&code))
6Z        {
p79          switch (class)
8b           {
f3C             case MOUSEBUTTONS:
bmE                ModifyIDCMP(wind,iflags);
6w                 return();
2fC             case MOUSEMOVE:
eYF                mox=wind->MouseX;moy=wind->MouseY;
wP                 SetMPoint(mox/8,(moy-11)/4,mode);
Jo9          }
Kp6       }
Lq3    }
Mr0 }
25  /*h2.9---------- Zeichen aus Zeichensatz in Zwischenspeiche
    r -----------*/
ky  void Font_to_Mem(ascii)
hy  long ascii;
Mp  {
ao  ULONG *adr=cmatrix+100*ascii;
nq  UBYTE *fadr=tfuser->tf_CharData;
e7  USHORT *loc=tfuser->tf_CharLoc;
gX  register long hohe,breite,offset,bitoff;
MR  register long i,j;
Qq  BYTE TestBit();
6r  void SetBit(),AsciiOut();
N23    if(ascii<=tfuser->tf_HiChar && ascii>=tfuser->tf_LoC
    har)
Vy     {
lr6       for(i=0;i<=99;i++)adr[i]=0;
5R        offset=ascii-tfuser->tf_LoChar;
tB        if(offset<0)offset=0;
Ju        hohe=tfuser->tf_YSize-1;
N3        bitoff=loc[offset*2];
Jf        breite=loc[offset*2+1]-1;
aR        if(hohe<0 || hohe>51) hohe=8;
Xi        if(breite<0 || breite>64)return();
i0        for(i=0;i<=hohe;i++)
f8        {
iJ9          for(j=0;j<=breite;j++)
hA           {
2LC             if(TestBit(fadr,j+bitoff))
RsF                SetBit(adr,j);
oJ9          }
F1           fadr+=tfuser->tf_Modulo;adr+=2;
qL6       }
rM3    }
n3     Ascii=ascii;
gZ     AsciiOut();
uP0 }
sQ  /*h2.10------------------ Zeichen Manipulieren ------------
    ---------------*/
1C  void Delete(mode)
1u  BYTE mode;
uN  {
ru  ULONG *adr=cmatrix+100*Ascii,*point=cmemo;
rw  register long i,j;
An3    switch(mode)
yR     {
qB6       case 0: /* CLR */
L99          switch(DMode)
1U           {
6MC             case 0:
mZF                for(i=0;i<=(YMatrix-1);i++,adr+=2)
4X                 {
bMI                   for(j=0;j<=(XMatrix-1);j++)
6Z                    {
WmL                      ClrBit(adr,j);
ChI                   }
DiF                }
9IC             break;
HY              case 1:
RXF                for(i=0;i<=99;i++)adr[i]=0;
Hm9          }
DM6       break;
5S        case 1: /* Snapshot */
bP9          switch(DMode)
Hk           {
McC             case 0:
MIF                for(i=0;i<=(YMatrix-1);i++,adr+=2,point+=2)

Kn                 {
rcI                   for(j=0;j<=(XMatrix-1);j++)
Mp                    {
UcL                      if(TestBit(adr,j))
QkO                         SetBit(point,j);
I5L                      else
9IO                         ClrBit(point,j);
VOI                   }
W1F                }
SbC             break;
ar              case 1:
LcF                for(i=0;i<=99;i++,point++)*point=adr[i];
VeC             break;
b69          }
XD           DrawLittle(1,0,0,cmemo);
Yh6       break;
Cs        case 2: /* Copy */
wk9          switch(DMode)
c5           {
hxC             case 0:
hdF                for(i=0;i<=(YMatrix-1);i++,adr+=2,point+=2)

f8                 {
CxI                   for(j=0;j<=(XMatrix-1);j++)
hA                    {
```

Listing 4. (Fortsetzung)

```
                    if(TestBit(point,j))
                       SetBit(adr,j);
                    else
                       ClrBit(adr,j);
                 }
              }
           break;
           case 1:
              for(i=0;i<=99;i++,adr++)*adr=point[i];
           break;
        }
     break;
  }
  Mem_to_Matrix();
}
/*h2.11------------------- Scrolling ausführen -----------------------*/
void Scroll(mode)
BYTE mode;
{
ULONG *adr=cmatrix+100*Ascii,*point;
long i,j,bitoff;
BYTE astep,bstep,cstep;
long xstart,xend,ystart,yend;
ULONG class;
USHORT code;
  switch(Quadrant)
  {
     case 0:
        xstart=0;ystart=0;xend=HelpLine;yend=BaseLine;
        break;
     case 1:
        xstart=HelpLine;ystart=0;xend=XMatrix;yend=BaseLine;
        break;
     case 2:
        xstart=0;ystart=BaseLine;xend=HelpLine;yend=YMatrix;
        break;
     case 3:
        xstart=HelpLine;ystart=BaseLine;xend=XMatrix;yend=YMatrix;
        break;
  }
  switch (mode)
  {
     case 0:
        astep=1;bstep=0;cstep=0;
        break;
     case 1:
        astep=-1;bstep=0;cstep=0;
        break;
     case 2:
        astep=0;bstep=0;cstep=64;yend--;
        break;
     case 3:
        astep=0;bstep=64;cstep=0;yend--;
        break;
  }
  /*--- Scroll left & right up & down ---*/
     adr=cmatrix+100*Ascii;
     for(i=0;i<=99;i++,adr++)scmemo[i]=*adr;
     adr=cmatrix+100*Ascii+2*ystart;
     point=scmemo+2*ystart;
     for(i=ystart;i<=(yend-1);i++)
     {
        for(j=xstart;j<=(xend-1);j++)
        {
           if(TestBit(point,j+astep+cstep))
              SetBit(adr,j+bstep);
           else ClrBit(adr,j+bstep);
        }
        switch (mode)
        {
           case 0: ClrBit(adr,xend);break;
           case 1: ClrBit(adr,xstart);break;
        }
        adr+=2;point+=2;
     }
     switch (mode)
     {
        case 2:
           adr=cmatrix+100*Ascii+2*yend;
           for(i=xstart;i<=(xend-1);i++)
              ClrBit(adr,i);
           break;
        case 3:
           adr=cmatrix+100*Ascii+2*ystart;
           for(i=xstart;i<=(xend-1);i++)
              ClrBit(adr,i);
     }
     Mem_to_Matrix();
}
/*h2.12---------- Zeichen aus Zwischenspeicher in Matrix ------------*/
void Mem_to_Matrix()
{
register long *adr;
register long i,j;
register UBYTE *point=drawmem;
  adr=cmatrix+100*Ascii;
  BltClear(drawmem,12800,1);
  for(i=0;i<=49;i++,point+=256)
  {
     for(j=0;j<=31;j++)
     {
        if(*adr & maske[j])
        {
           *(point+64+j)=127;
           *(point+128+j)=127;
           *(point+192+j)=127;
        }
     }
     adr++;
     for(j=32;j<=63;j++)
     {
        if(*adr & maske[j-32])
        {
           *(point+64+j)=127;
           *(point+128+j)=127;
           *(point+192+j)=127;
        }
     }
     adr++;
  }
  point=drawmem;
  drima->ImageData=point;
  DrawImage(rast,drima,0,0);
  DrawLittle(0,0,0,cmatrix+100*Ascii);
  Mabox(1);
}
/*------------------------------------------------------------------------*/
(C) 1987 M&T
```

Listing 4. (Schluß)

Programmname: h3.c

Computer: siehe Listing 1

```
Programm : h3.c
---------------

/*---(FontDesigner)------------ File h3.c ------------*/
#include <exec/memory.h>
#include <intuition/intuition.h>
#include <intuition/intuitionbase.h>
#include <libraries/dos.h>
#include <graphics/gfxmacros.h>
#include "header.h"
/*--------------------- Externe Variablen --------------------------*/
extern struct Image *oima,*drima,*mima;
extern struct FileInfoBlock *f_info;
extern struct RastPort *rast;
extern struct Window *wind;
extern long XMatrix,YMatrix,BaseLine,HelpLine,Ascii,DMode;
extern long Quadrant,undoascii;
extern ULONG *cmatrix;
```

```
QL  extern BYTE *drawmem;
Tl  extern ULONG cmemo[],scmemo[],undomem[];
hg  /*------------------- Daten für Programmodus -----------------------*/
3U  short pdat[][4]=
GJ  {
kW3     { 10,16,88,28 }, { 105,16,167,28 },{ 184,16,254,28 },
K6      { 10,33,56,45 },{ 76,33,122,45 },{ 142,33,188,45 },{ 208,33,254,45 },
wD      { 10,50,56,62 },{ 76,50,122,62 },{ 142,50,164,62 },{ 172,50,194,62 },
by      { 202,50,224,62 },{ 232,50,254,61 },
mf      { 30,67,50,77 },{ 30,87,50,97 },{ 10,77,30,87 },{ 50,77,70,87 },
Y8      { 80,76,118,88 },{ 128,76,166,88 },
24      { 176,76,254,88 },
1h0 };
UJ  char *ptxt[]=
Qt  {
Q33     "START:","END:","STEP:"," INV"," ROT","MIR_H","MIR_B","EXP_X","EXP_Y",
DB      "Q1","Q2","Q3","Q4","UP","DOWN","LEFT","RIGHT","DELE","GO","PZ:",
qm0 };
WM  short parr[][7]=
Vy  {
tT3     { 40,68,47,75,33,75,-2 },
bM      { 40,96,47,89,33,89,-2 },
1Z      { 13,82,25,79,25,86,-2 },
x1      { 67,82,55,79,55,86,-2 },
xt0 };
1h  char programm[100];
md  /*h3.1------------------------ Zeichen spiegeln -----------------------*/
dr  void Mirror(mode)
mf  BYTE mode;
f8  {
57  register long i,j,*adr=cmatrix+100*Ascii,*point;
2b  register long bitoff;
vY3    switch(mode)
JC     {
o46       case 0:
sp9          for(i=0;i<=YMatrix;i++,adr+=2)
mF           {
s0C             for(j=0;j<=HelpLine;j++)
oH              {
w4F                if(TestBit(adr,j))
92I                   SetBit(adr,2*HelpLine-j-1);
vQC             }
wR9          }
s1           break;
0H6       case 1:
Hn9          bitoff=2*64*(BaseLine-1)+64;point=adr;
zF           for(i=0;i<=BaseLine;i++,adr+=2,bitoff-=64)
xQ           {
vMC             for(j=0;j<=XMatrix;j++)
zS              {
7FF                if(TestBit(adr,j))
3dI                   SetBit(point,bitoff+j);
6bC             }
7c9          }
3C           break;
9e3    }
FT     Mem_to_Matrix();
Bg0 }
I5  /*h3.2-------------- Undospeicher in Hauptspeicher -----------------*/
JM  void Undo()
Ad  {
zb  register long i,*adr=cmatrix+100*Ascii;
4c3    for(i=0;i<=99;i++,adr++)*adr=undomem[i];
6z     Ascii=undoascii;
5y     AsciiOut();
0c     Mem_to_Matrix();
Kp0 }
BZ  /*h3.3--------------------- Zeichen drehen ------------------------*/
Zx  void Rotate()
Jm  {
Ni  register long *adr=cmatrix+100*Ascii,*point=scmemo;
zK  register long i,j,bitoff;
CT3    for(i=0;i<=99;i++,point++)*point=adr[i];
B0     point=scmemo;
cw     for(i=0;i<=XMatrix;i++,point+=2)
Ps     {
WK6       bitoff=(YMatrix-1)*64+i;
zB        for(j=0;j<=YMatrix;j++,bitoff-=64)
Sv        {
wu9          if(TestBit(point,j))
aaC             SetBit(adr,bitoff);
7j9          else ClrBit(adr,bitoff);
a56       }
b63    }
hv     Mem_to_Matrix();
d80 }
BZ  /*h3.4----------------------- Zeichen invertieren -----------------------*/
6m  void Revers()
c5  {
ey  register long i,j,*adr=cmatrix+100*Ascii,xend=XMatrix,yend=YMatrix;
Nj3    for(i=0;i<=(yend-1);i++,adr+=2)
f8     {
Sv6       for (j=0;j<=(xend-1);j++)
hA        {
px9          if(TestBit(adr,j))
80C             ClrBit(adr,j);
dQ9          else
TuC             SetBit(adr,j);
qL6       }
rM3    }
xB     Mem_to_Matrix();
t00 }
Ay  /*h3.5---------- Zeichen in x-Richtung vergrößern -------------------*/
Yt  void Expand_x()
sL  {
z2  register ULONG *adr=cmatrix+100*Ascii,*point=scmemo;
pu  register long i,j;
123    for(i=0;i<=99;i++,point++)*point=adr[i];
kx     point=scmemo;
ie     for(i=0;i<=YMatrix-2;i++,adr+=2,point+=2)
yR     {
kX6       for(j=0;j<=XMatrix-1;j++)
0T        {
US9          if(TestBit(point,j))
2V           {
40C             SetBit(adr,j*2);SetBit(adr,j*2+1);
8d9          }
y1           else
6Z           {
LJC             ClrBit(adr,j*2);ClrBit(adr,j*2+1);
Ch9          }
D16       }
EJ3    }
KY     Mem_to_Matrix();
G10 }
k6  /*h3.6------------- Zeichen in y-Richtung vergrößern --------------*/
zL  void Expand_y()
F1  {
0Q  ULONG *adr=cmatrix+100*Ascii,*point=scmemo;
M3  register long i;
8P3    for(i=0;i<=99;i++,point++)*point=adr[i];
7K     point=scmemo;
I5     for(i=0;i<=99;i+=4)
Lo     {
aP6       adr[i]=*point;adr[i+2]=*point;point++;
ak        adr[i+1]=*point;adr[i+3]=*point;point++;
Sx3    }
Ym     Mem_to_Matrix();
Uz0 }
1W  /*h3.7--------------- Programmodus aktivieren ----------------------*/
t8  void Programm()
Tw  {
qZ  static short start=32,end=255,step=1,count=0;
Uk  ULONG flags=WINDOWCLOSE|ACTIVATE|WINDOWDRAG;
KA  ULONG iflags=CLOSEWINDOW|MOUSEBUTTONS;
aG  struct Window *wn;
uA  struct RastPort *rp;
0i  ULONG class;
Wj  USHORT code;
```

Listing 5. Geben Sie »h3.c« bitte mit dem Checksummer (Seite 159) ein

```
short i,mox,moy,flag=0;
char htext[10];
void pz_out(),Go_Amiga();
  wn=Make_Window(170,140,278,105,"Programm",flags,iflags,0);
  if(wn==0)return();
  rp=wn->RPort;
  SetAPen(rp,1);RectFill(rp,0,11,278,105);
  Box(rp,2,12,275,103,0,JAM1);
  for (i=0;i<=19;i++)
  {
     SetAPen(rp,0);
     RectFill(rp,pdat[i][0],pdat[i][1],pdat[i][2],pdat[i][3]);
     Box(rp,pdat[i][0],pdat[i][1],pdat[i][2],pdat[i][3],2,JAM2);
     Box(rp,pdat[i][0]-1,pdat[i][1],pdat[i][2]+1,pdat[i][3],2,JAM2);
     if(i<13 || i>16)
       Text_out(rp,pdat[i][0]+3,pdat[i][1]+3,3,0,JAM2,ptxt[i],0);
     else
       FDrawPolygon(rp,parr[i-13],3,JAM1);
  }
  pz_out(rp,start,end,step);
  if(count==0)
     Text_out(rp,203,79,3,0,JAM2,"NULL",0);
  else
     Text_out(rp,203,79,3,0,JAM2,ptxt[programm[count-1]],0);
  FOREVER
  {
     while(ExaIntui(wn,&class,&code))
     {
        mox=wn->MouseX;moy=wn->MouseY;
        switch (class)
        {
           case CLOSEWINDOW:
              CloseWindow(wn);
              return();
           case MOUSEBUTTONS:
              if(code != SELECTDOWN) break;
              i=Prurec(mox,moy,pdat,0,19);
              if(i<0)break;
              if(i>2 && i<18)
              {
                 if(flag==0)count=0;
                 Invert(rp,pdat[i]);
                 Delay(5);
                 programm[count]=i;
                 programm[count+1]=0;
                 sprintf(htext,"%s     ",ptxt[i]);
                 Text_out(rp,203,79,3,0,JAM2,htext,5);
                 count++;
                 if(count>98)count--;
                 flag=1;
              }
              SetAPen(rp,1);
              switch (i)
              {
                 case 0:
                    start=(short)input_zahl(wn,61,19,3,"");
                    if(start<0 || start>255)start=32;
                    break;
                 case 1:
                    end=(short)input_zahl(wn,140,19,3,"");
                    if(end<start || end>255)end=255;
                    break;
                 case 2:
                    step=(short)input_zahl(wn,227,19,3,"");
                    if(step<1 || step>255)step=1;
                    break;
                 case 18:
                    Text_out(rp,pdat[i][0]+3,pdat[i][1]+3,1,0,JAM2,
                                "STOP",0);
                    Invert(rp,pdat[i]);
                    Go_Amiga(wn,start,end,step);
                    break;
                 case 19:
                    Text_out(rp,203,79,3,0,JAM2," NULL",0);
                    Invert(rp,pdat[i]);
                    count=0;programm[0]=0;
                    break;
              }
              if(i>2)Invert(rp,pdat[i]);
              Text_out(rp,pdat[18][0]+3,pdat[18][1]+3,3,0,JAM2,
                  " GO ",0);
              pz_out(rp,start,end,step);
        }
     }
  }
}
/*---- Unterfunktion zu h3.7 ----*/
void pz_out(rp,st,end,stp)
struct RastPort *rp;
short st,end,stp;
{
char htext[10];
  sprintf(htext,"%3.0f",(float)st);
  Text_out(rp,61,19,3,0,JAM2,htext,0);
  sprintf(htext,"%3.0f",(float)end);
  Text_out(rp,140,19,3,0,JAM2,htext,0);
  sprintf(htext,"%3.0f",(float)stp);
  Text_out(rp,227,19,3,0,JAM2,htext,0);
}
/*---- Unterfunktion zu h3.7 ----*/
void Go_Amiga(wn,st,end,stp)
struct Window *wn;
short st,end,stp;
{
short i,count;
long mer1=Ascii,mer2=Quadrant;
ULONG class;
USHORT code;
  for(i=st;i<=end;i+=stp)
  {
     Ascii=i;count=0;AsciiOut();
     ExaIntui(wn,&class,&code);
     if(class==MOUSEBUTTONS && code==SELECTDOWN)break;
     while (programm[count] != 0)
     {
        Invert(wn->RPort,pdat[programm[count]]);
        switch (programm[count])
        {
           case 3: Revers();break;
           case 4: Rotate();break;
           case 5: Mirror(0);break;
           case 6: Mirror(1);break;
           case 7: Expand_x();break;
           case 8: Expand_y();break;
           case 9: case 10: case 11: case 12:
              Quadrant=programm[count]-9;
              break;
           case 13: Scroll(2);break;
           case 14: Scroll(3);break;
           case 15: Scroll(0);break;
           case 16: Scroll(1);break;
           case 17: Delete(0);break;
        }
        Invert(wn->RPort,pdat[programm[count]]);
        count++;
     }
  }
  Ascii=mer1;Quadrant=mer2;
  Mem_to_Matrix();AsciiOut();
}
/*h3.8--------------- Gesamten Zeichensatz anzeigen --------------*/
void Show()
{
ULONG class;
USHORT code;
short Font_out(),ascii=0;
  Mabox(0);
  ascii=Font_out(ascii);
  FOREVER
  {
     while(ExaIntui(wind,&class,&code))
     {
        switch (code)
        {
           case MENUDOWN:
              if(ascii>=254)ascii=0;
              ascii=Font_out(ascii);
```

```
7G                  break;
GpC             case SELECTDOWN:
JXF                 Mem_to_Matrix();
pF                  Mabox(1);
7x                  return();
Hm9             }
In6         }
Jo3     }
Kp0 }
8G  /*----- Unterfunktion zu h3.8 -----*/
HG  short Font_out(ascii)
03  short ascii;
Kn  {
73  register long xa=4,ya=13;
Jm3     SetAPen(rast,0);SetDrMd(rast,JAM1);
4d      RectFill(rast,1,11,511,210);
GS      while( ya<160 && ascii<254)
Ps      {
Z56         xa=4;
nL          while(xa<447 && ascii<254)
Sv          {
k39             DrawLittle(0,xa-522,ya-15,cmatrix+100*ascii);
jF              Box(rast,xa-1,ya-1,xa+XMatrix+1,ya+YMatrix+1,2,JAM
                1);
aJ              ascii++;
RO              xa+=XMatrix+2;
b66         }
b7          ya+=YMatrix+2;
d83     }
11      return(ascii);
fA0 }
kb  /*h3.9----------------- Speicher besorgen -----------------
    ----*/
t4  BYTE GetMemory(mode)
mf  BYTE mode;
f8  {
Aw  register BYTE *bpoi;
if  register short i,j;
Yp3     if(mode==0) goto cleanup1;
Me      oima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR)
        ;
ps      if(oima==0)return(-1);
1F      drima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR
        );
SH      if(drima==0)goto cleanup6;
Mc      mima=AllocMem(sizeof(struct Image),MEMF_CHIP|MEMF_CLEAR)
        ;
8r      if(mima==0)goto cleanup5;
9L      f_info=AllocMem(sizeof(struct FileInfoBlock),MEMF_CLEAR)
        ;
dD      if(f_info==0)goto cleanup4;
IL      cmatrix=AllocMem(102400,MEMF_CLEAR);
00      if(cmatrix==0) goto cleanup3;
jn      drawmem=AllocRaster(512,400);
yG      if(drawmem==0) goto cleanup2;
7s      oima->LeftEdge=522;oima->TopEdge=15;oima->Width=64;oi
        ma->Height=50;
aU      oima->Depth=1;oima->ImageData=NULL;oima->PlanePick=1;

Gu      oima->PlaneOnOff=0;oima->NextImage=NULL;
di      drima->LeftEdge=0;drima->TopEdge=11;drima->Width=512;
        drima->Height=200;
P2      drima->Depth=2;drima->ImageData=NULL;drima->PlanePick
        =3;
tM      drima->PlaneOnOff=0;drima->NextImage=NULL;
r0      mima->LeftEdge=522;mima->TopEdge=71;mima->Width=64;mi
        ma->Height=50;
MK      mima->Depth=1;mima->ImageData=NULL;mima->PlanePick=1;

6q      mima->PlaneOnOff=0;mima->NextImage=NULL;
H6      bpoi=drawmem+12800;
2F      for (i=0;i<=199;i++)
6Z      {
KQ6         for(j=0;j<=63;j++,bpoi++)
8b          {
nY9             *bpoi=128;
fc              if(i%4 == 0)
2JC                 *bpoi=0xff;
G16         }
Hm3     }
1g      return(0);
cD0 cleanup1: FreeRaster(drawmem,512,400);
hP  cleanup2: FreeMem(cmatrix,102400);
QB  cleanup3: FreeMem(f_info,sizeof(struct FileInfoBlock));
R3  cleanup4: FreeMem(mima,sizeof(struct Image));
Wp  cleanup5: FreeMem(drima,sizeof(struct Image));
1R  cleanup6: FreeMem(oima,sizeof(struct Image));
S33     return(-1);
Qv0 }
dj  /*h3.11------ Sicherheitsabfragen un Meldungen ausgeben ---
    -----*/
1j  char *shtxt[]=
Ps  {
IX3     "Are you shure to quit ?",
PW      "I can't find this font !",
mp      "ERROR !! Font not saved.",
IO      "Not enough Memory !!!",
Co      "I can't find Diskfont Library",
Ai      "No ASSIGN ! Please use CLI",
53      "Loading Font !  Please Wait",
EM      "Loading Topaz/8 !  Please Wait",
q8      "Examine Font !!",
WM      "One Moment ! Writing Char Data !",
24      "Creat New Fastdirectory !",
50      "Writing xxxx.font !",
zv0 };
qp  BYTE Shure(nr)
83  BYTE nr;
f8  {
WS  static flag=0;
GZ  static struct Window *swind;
C3  ULONG flags=ACTIVATE|WINDOWDRAG;
Ys  ULONG class;
gt  USHORT code;
DJ  char text[2][4];
fD  short sh_data[2][4];
YH  short wbr;
jF  BYTE i;
g23     if(nr>=6 && flag==1)
qJ      {
Pm6         flag=0;
se          CloseWindow(swind);
QL          return(0);
yT3     }
hN      wbr=strlen(shtxt[nr])*8+28;
JG      sh_data[0][0]=10;sh_data[0][1]=32;sh_data[0][2]=40;sh_da
        ta[0][3]=44;
9k      if(nr>0) sh_data[0][2]=32;
ef      sh_data[1][0]=wbr-44;sh_data[1][1]=32;
ft      sh_data[1][2]=wbr-22;sh_data[1][3]=44;
r8      if(nr<1)
1U      {
v06         strcpy(text[0],"YES");strcpy(text[1],"NO");
7c3     }
xk      else
5Y      {
FS6         strcpy(text[0],"OK");strcpy(text[1],text[0]);
Bg3     }
AP      swind=Make_Window(0,0,wbr,51,"REQUEST",flags,MOUSEBUTTON
        S,0);
13      if(swind==0)return(0);
TM      SetAPen(swind->RPort,1);RectFill(swind->RPort,0,11,wbr
        ,51);
DA      Box(swind->RPort,2,12,wbr-3,49,0,JAM1);
KO      Text_out(swind->RPort,10,16,0,1,JAM2,shtxt[nr],0);
Jf      if(nr<6)
Eh      {
gK6         for(i=0;i<=1;i++)
GJ          {
6I9             Box(swind->RPort,sh_data[i][0],sh_data[i][1],sh_d
                ata[i][2],
hbN                         sh_data[i][3],2,JAM1);
8a9             Text_out(swind->RPort,sh_data[i][0]+3,sh_data[i][
                1]+3,0,1,JAM2,
icI                     text[i],0);
Pu6         }
Qv3     }
G3      else
Or      {
zN6         flag=1;
xs          return(0);
V03     }
41      FOREVER
Tw      {
Ep6         while(ExaIntui(swind,&class,&code))
Vy          {
oH9             if(code!=SELECTDOWN)break;
Ju              i=Prurec(swind->MouseX,swind->MouseY,sh_data,0,1
                );
```

Listing 5. (Fortsetzung)

```
RH              if(i<0)break;
ZL              CloseWindow(swind);
nd              return(i);
fA6         }
gB3     }
hC0 }
J0  /*h3.12---------------------- Direktory sortieren ---------------------*/
qT  void SortDir(names,anz)
62  BYTE *names;
oV  short anz;
iB  {
Qk  register short i,j=1;
Ip  BYTE *p1=names,*p2,*p3,h;
U93     p3=AllocMem(1024,MEMF_CLEAR);
nB      if(p3==0)return();
Lt      p2=p3;
jE      for(i=0;i<=1023;i++,p1++)p2[i]=*p1;
6K      p1=names;
W6      for(i=1;i<=anz-1;i++)
rK      {
E46         h=*p2;p2++;
Am          if(h==1)
uN          {
vv9             *p1=1;p1++;
Ln              strcpy(p1,p2);
8p              p1+=strlen(p2)+1;
2X6         }
Dv          p2+=strlen(p2)+1;
4Z3     }
Z7      p2=p3;
iI      for(i=1;i<=anz-1;i++)
3W      {
QG6         h=*p2;p2++;
Ju          if(h==0)
6Z          {
159             *p1=0;p1++;
Xz              strcpy(p1,p2);
K1              p1+=strlen(p2)+1;
Ej6         }
P7          p2+=strlen(p2)+1;
G13     }
pw      FreeMem(p3,1024);
In0 }
SL  /*h3.13--------------- Zeichensatzdiskette aktualisieren ----------------*/
yS  void Assign(drive)
Ro  char *drive;
I1  {
eC  struct FileLock *lockp,*Lock();
OT  char text[100],*point,i;
sm3     strcpy(text,drive);text[4]=0;
vk      if((lockp=Lock(text,ACCESS_READ))==0)
I86         return();
3H3     if(Examine(lockp,f_info)==0)
KA6         return();
He3     UnLock(lockp);
fi      /*-- Stern und Anführungszeichen im Filenamen bearbeiten --*/
iV      point=f_info->fib_FileName;
FO      strcpy(text,f_info->fib_FileName);
T8      for(i=0;i <= strlen(text);i++,point++)
Vy      {
jw6         *point=text[i];
zt          if(text[i]=='*' || text[i]=='"')
Y1          {
oc9             *point='*';point++;
n0              *point=text[i];
fA6         }
gB3     }
sg      /*-----------------------------------*/
3k      sprintf(text,"assign > nil: fonts: \"%s:%s\"",
byB             f_info->fib_FileName,drive+4);
7d3     Execute(text,0,0);
1G0 }
YM  /*------------------------------------------------------------------*/
```

(C) 1987 M&T

Listing 5. (Schluß)

Amiga kunterbunt

Daß der Amiga mit Farben nicht gerade geizt, ist allgemein bekannt. Aber wie steht es um Funktionen zur Bearbeitung und Manipulation bestimmter Farben in einer Grafik? Wir haben genau auf diesem Bereich ein Defizit entdeckt, das mit »BlitColor« behoben wird.

Die Grafik-Bibliothek des Amiga stellt umfangreiche Funktionen zur Verfügung, einzelne Bildteile zu kopieren und sie gleichzeitig zu verfremden. Allerdings kommt manchmal der Wunsch auf, dieses auch auf einzelne Farben zu beschränken. Nach einer ergebnislosen Suche in der »graphics.library« ist der C-Programmierer dann auf sich selbst angewiesen. Steht man mit dem Blitter eventuell auf Kriegsfuß, dann sieht das Ganze ziemlich hoffnungslos aus.

Die hier vorgestellten Funktionen des Programms »BlitColor« (Listing 1) sollen nun helfend eingreifen und die Grundlage für das »farbweise« Bearbeiten von Bildern schaffen. Dies ist einigen Anwendern zum Beispiel durch die Programme »PixMate« oder »Butcher« bekannt.

Was steckt hinter der Maske?

Um die Funktionen verstehen zu können, muß zunächst einmal der Begriff »Maske« näher erläutert werden. Eine Maske ist, wenn man so will, eine Art Tabelle, in der vermerkt wird, ob an einer bestimmten Position ein Zustand auftritt oder nicht. Auf eine spezielle Farbe einer Grafik bezogen gibt die Maske also einen Hinweis, ob diese Farbe vorhanden ist oder nicht. Möchte man beispielsweise für einen 320 x 200 großen und vier Bitplanes tiefen Screen eine Maske erstellen, so bleibt nichts anderes übrig, als bei jedem Punkt »nachzuschauen«, ob er die entsprechende Farbe hat oder nicht. Dieses wird dann mit »1« oder »0« in die Tabelle eingetragen. Demnach muß die Maske auch genau die Größe unserer Farbgrafik haben, also 320 x 200 Bit. Es ist dabei auch egal, wie »tief« unsere Grafik ist, denn die Maske enthält ja nur die Information über das Auftreten einer einzigen Farbe. Soll nur ein Ausschnitt aus dem Screen benutzt werden, so kann die Maske entsprechend kleiner sein. Letztendlich ist eine Maske mit einer Grafik gleichzusetzen, die aber immer nur eine Bitplane benötigt.

Um nun eine Maske dieser Art zu erstellen, wird die Funktion »GetColorMask« aufgerufen. Sie brauchen nur die notwendigen Parameter zu übergeben, die folgendermaßen definiert sind:

```
void GetColorMask (SourceBitMap,SourceX,SourceY,
MaskBitMap,MaskX,MaskY,
SizeX,SizeY, Color)
struct BitMap *Source
BitMap, *MaskBitMap;
WORD MaskX,MaskY,SizeX,
SizeY; UBYTE Color;
```

»SourceBitMap« gibt die Bitmap der Grafik an, in der Sie Veränderungen vornehmen wollen. Diese muß natürlich entsprechend ermittelt oder initialisiert werden.

»MaskBitMap« ist der Grafikbereich, den Sie als Maske benutzen wollen. Auch diese Bitmap-Struktur sollte richtig initialisiert sein. »SourceX« und »SourceY« legen fest, ab welcher X- und Y-Position mit der Maskierung begonnen werden soll.

»MaskX« und »MaskY« bewirken, daß die maskierten Daten auch innerhalb der Maskbitplane verschiedene Positionen annehmen können. Die Größe des Bereichs wird mit »SizeX« und »SizeY« eingestellt, und die Farbe wird letztendlich mit »Color« ausgewählt.

Eine Farbe wird ersetzt...

Haben Sie erst einmal die Maske für einen bestimmten Bereich in der Grafik ermittelt, so läßt sich diese nun dazu verwenden, die festgelegten Punkte durch eine andere Farbe zu ersetzen. Dazu rufen Sie die Funktion »CreateColor« auf. Im Vergleich zu »GetColorMask« sind nur die beiden Bitmap-Parameter vertauscht, da ja nun der umgekehrte Vorgang stattfindet.

Mit anderen Worten: Ist in der Maskbitplane ein Pixel gesetzt, so bekommt der entsprechende Pixel in der Destinationbitplane — ehemals Sourcebitplane — die mit Color ausgewählte Farbe; ist der Maskbitplane-Pixel gelöscht, so bleibt der Pixel in der Destinationbitmap unverändert. Alle weiteren Parameter sind identisch. Dieser Vorgang läßt sich mit der Routine »ReplaceColor« auch auf einmal erledigen; es müssen nur zwei Farben angegeben werden, nämlich die zu ersetzende und die neue Farbe.

...oder ausgetauscht

Möchten Sie nun zwei Farben austauschen, so ist es unumgänglich, sich eine zweite MaskBitMap anzulegen, da nun das Auftreten von zwei Farben vermerkt werden muß. Daher benötigt die Funktion »ExchangeColor« drei weitere Argumente. Dies ist der Zeiger auf eine weitere Bitmap-Struktur und die Werte der X- und Y-Koordinate in der zweiten Maskbitplane. Die beiden auszutauschenden Farben werden notwendigerweise auch angegeben.

Wenn sich der Guru meldet

Innerhalb der Funktionen erfolgt keine Prüfung der übergebenen Argumente. Die Funktionen sind daher recht empfindlich gegenüber falsch übergebenen Parametern und rächen sich meist mit einem verwüsteten Speicher. Um ganz sicher zu gehen, sollten Sie die Maskbitplane immer auf die Größe der zu bearbeitenden Bitplanes bringen, so daß wirklich jeder Bereich manipuliert werden kann. Wer mit dem Gedanken spielt, sich seine Maskbitplanes selbst anzulegen, der sollte bedenken, daß diese im Chip-Memory liegen müssen. Unsere Routinen greifen nämlich auf die Funktion »BlitBitMap« der »graphics.library« zurück, die den Blitter benötigt. Dieser wiederum gehört zu den Customchips, die nur die untersten 512K des Speichers adressieren können, also das sogenannte Chip-Memory.

BitMap ist nicht gleich Rastport

Sie werden sich bestimmt schon gewundert haben, warum bisher nur Bitmaps zur Festlegung der Grafik benutzt wurden und nicht die dafür sonst häufig benutzten Rastports? Nun, das liegt einfach daran, daß Rastports die Bearbeitung von überlappenden Speicherbereichen innerhalb der Bitplanes ermöglichen, das sogenannte »Clipping«. Dies tritt zum Beispiel auf, wenn Sie in der Workbench ein Fenster auf ein anderes legen und diese sich somit teilweise oder ganz überlagern. Zur Verwaltung der daraus resultierenden »Schnittspeicherbereiche« wird die »layer.library« benutzt, was natürlich zusätzlichen Rechenaufwand erfordert und Grafikoperationen bremst. Unsere Routinen arbeiten mit Bitmaps und sind daher sehr schnell.

Möchten Sie allerdings als Programmierer die Möglichkeit des »Clippings« nicht missen, so wird einfach in unserem C-Quelltext die Funktion »BlitBitMap« durch »ClipBlit« ausgetauscht. Diese führt im Prinzip die gleiche Operation durch, jedoch auf Rastports bezogen. Ebenso muß vorher die Angabe der zu verarbeitenden Bitplanes — das zehnte Argument beim BlitBitMap-Aufruf — durch den Makro »SetWrMsk()« ersetzt werden, da sich hierfür extra ein Element in der Rastport-Struktur befindet. Als letzter Schritt wird überall »BitMap« zu »RastPort« umbenannt; es soll ja mit RastPorts gearbeitet werden.

Unser Demo-Programm (Listing 2) generiert zunächst eine Zufallsgrafik in verschiede-

nen Farben. Mittels der Exchange-Funktion werden dann auf der linken Screenhälfte Farben vertauscht. Das Programm endet mit einer neuen Variante, ein Bild auszublenden: Die Farben werden mit »ReplaceColor« nacheinander zusammengerafft. Der Einfachheit halber werden hier für die beiden MaskBitPlanes zwei Screens geöffnet, die Sie sich während der Ausführung des Programms einmal anschauen sollten.

(Thomi Mauch/ Christian Wolf/rs)

Programmname:	BlitColor.h
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztek-C V3.4 und Lattice-C V4.0
Aufrufe:	abhängig vom verwendeteten Compiler
Bemerkung:	Die Routine ist in eigene C-Programme mit »#include« einzubinden

```
Programm : BlitColor.h

----------------------

 1 R50 /*******************************************************/
 2 Cq  /***                                               ***/
 3 7Z  /***        Routinen zum Bearbeiten von Farben      ***/
 4 Es  /***                                               ***/
 5 kp  /***          written in 1988 by Thomi Mauch        ***/
 6 Gu  /***                                               ***/
 7 XB  /*******************************************************/
 8 iU  struct BitMap TmpBitMap;
 9 qN  InitTmpBitMap (BitMap)
10 2W  struct BitMap *BitMap;
11 7a  {
12 zX3    /* interner Gebrauch: verbiegt Pointer einer BitMap */
13 3h     WORD i;
14 TG     TmpBitMap.BytesPerRow = BitMap->BytesPerRow;
15 SC     TmpBitMap.Rows = BitMap->Rows;
16 B2     TmpBitMap.Flags = BitMap->Flags;
17 2x     TmpBitMap.pad = BitMap->pad;
18 uR     TmpBitMap.Depth = 8;
19 Br     for (i=0; i<8; i++)
20 TG6       TmpBitMap.Planes[i] = BitMap->Planes[0];
21 Lq0 }
22 7k  GetColorMask (SrcBitMap,SrcX,SrcY,MaskBitMap,MaskX,MaskY,Siz
       eX,SizeY,Color)
23 Kx  struct BitMap *SrcBitMap,*MaskBitMap;
24 Im  WORD SrcX,SrcY,MaskX,MaskY,SizeX,SizeY;
25 d1  BYTE Color;
26 Mp  {
27 YE3    /*** extrahiert eine Farbe aus einer BitMap und ***/
28 Vn     /*** stellt sie auf einer einzigen BitPlane als ***/
29 UT     /*** Maske dar                                  ***/
30 Ky     WORD i;
31 xa     UBYTE Minterm;
32 Q1     InitTmpBitMap (MaskBitMap);
33 G3     if (Color & 1)
34 A76       Minterm = 0xc0;       /* Dest = Source */
35 OB3    else
36 ZV6       Minterm = 0x30;       /* Dest = NOT Source */
37 zs3    BltBitMap (SrcBitMap,SrcX,SrcY,&TmpBitMap,MaskX,MaskY,Siz
          eX,SizeY,Minterm,1,NULL);
38 N2     for (i=1; i<SrcBitMap->Depth; i++)
39 Z26       {
40 tm        if (Color & (1<<i))
41 E89          Minterm = 0x80;   /* Dest = Source AND Dest */
42 VI6       else
43 Tc9          Minterm = 0x20;   /* Dest = NOT Source AND Dest */
44 4h6       BltBitMap (SrcBitMap,SrcX,SrcY,&TmpBitMap,MaskX,MaskY,
             SizeX,SizeY,Minterm,1<<i,NULL);
45 jE        }
46 kF0 }
47 mm  CreateColor (MaskBitMap,MaskX,MaskY,DestBitMap,DestX,DestY,S
       izeX,SizeY,Color)
48 5V  struct BitMap *MaskBitMap,*DestBitMap;
49 JG  WORD MaskX,MaskY,DestX,DestY,SizeX,SizeY;
50 2Q  BYTE Color;
51 1E  {
52 tD3    /*** verarbeitet eine Maske zu einer Farbe und  ***/
53 xo     /*** kopiert sie auf das Ziel, ohne den Hinter- ***/
54 ZI     /*** grund zu verändern                         ***/
55 JN     WORD i;
56 Mz     UBYTE Minterm;
57 pA     InitTmpBitMap (MaskBitMap);
58 WP     for (i=0; i<DestBitMap->Depth; i++)
59 tM6       {
60 D6        if (Color & (1<<i))
61 129          Minterm = 0xe0;   /* Dest = Source OR Dest */
62 pc6       else
63 nw9          Minterm = 0x20;   /* Dest = NOT Source AND Dest */
64 F86       BltBitMap (&TmpBitMap,MaskX,MaskY,DestBitMap,DestX,Des
             tY,SizeX,SizeY,Minterm,1<<i,NULL);
65 3Y        }
66 4Z0 }
67 jT  ReplaceColor (BitMap,X,Y,MaskBitMap,MaskX,MaskY,SizeX,SizeY,
       FromColor,ToColor)
68 GY  struct BitMap *BitMap,*MaskBitMap;
69 Kz  WORD X,Y,MaskX,MaskY,SizeX,SizeY;
70 3Y  BYTE FromColor,ToColor;
71 5Y  {
72 T73    /*** Ersetzt eine Farbe durch eine andere       ***/
73 AU     GetColorMask (BitMap,X,Y,MaskBitMap,MaskX,MaskY,SizeX,Siz
          eY,FromColor,NULL);
74 Zd     CreateColor (MaskBitMap,MaskX,MaskY,BitMap,X,Y,SizeX,Size
          Y,ToColor,NULL);
75 D10 }
76 ZF  ExchangeColor (BitMap,X,Y,Mask0BitMap,Mask0X,Mask0Y,Mask1Bit
       Map,Mask1X,Mask1Y,SizeX,SizeY,Color0,Color1)
77 B4  struct BitMap *BitMap,*Mask0BitMap,*Mask1BitMap;
78 Br  WORD X,Y,Mask0X,Mask0Y,Mask1X,Mask1Y,SizeX,SizeY;
79 V9  BYTE Color0,Color1;
80 Eh  {
81 dd3    /*** Vertauscht zwei Farben                     ***/
82 PO     GetColorMask (BitMap,X,Y,Mask0BitMap,Mask0X,Mask0Y,SizeX,
          SizeY,Color0,NULL);
83 Jh     GetColorMask (BitMap,X,Y,Mask1BitMap,Mask1X,Mask1Y,SizeX,
          SizeY,Color1,NULL);
84 5o     CreateColor (Mask0BitMap,Mask0X,Mask0Y,BitMap,X,Y,SizeX,S
          izeY,Color1,NULL);
85 Fv     CreateColor (Mask1BitMap,Mask1X,Mask1Y,BitMap,X,Y,SizeX,S
          izeY,Color0,NULL);
86 Ot0 }
(C) 1987 M&T
```

Listing 1. Die Routine »BlitColor.h« binden Sie einfach mit »#include« in eigene Programme ein. Bitte mit dem Checksummer (Seite 159) eingeben.

Programmname:	Demo.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	s. Listing 1
Aufrufe:	s. Listing 1
Bemerkung:	Demo zeigt einige Möglichkeiten der Routine »BlitColor«

```
Programm : Demo.c

----------------

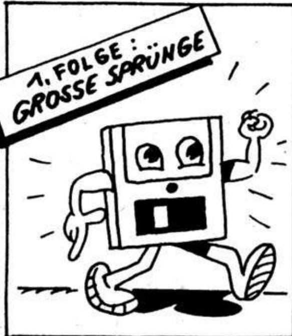
 1 ks0 #include "exec/types.h"
 2 Y1  #include "intuition/intuition.h"
 3 NZ  #include "graphics/text.h"
 4 Up  #include "graphics/gfx.h"
 5 X5  #include "BlitColor.h"
 6 Zs  struct Screen *OpenScreen();
 7 45  LONG GfxBase,
 8 Ey5      IntuitionBase;
 9 RhO struct RastPort *rp;
10 13  struct Screen *screen[3];
11 og  struct NewScreen ns =
12 8b3    {
13 3P     0,0,320,256,0,
14 KP     0,1,0,
15 B6     CUSTOMSCREEN
16 ZV     };
17 Wn0 static UWORD color [32] =
18 Eh3    {
19 gm     0x000, 0xfff, 0xff0, 0xfc0, 0xf90, 0xf60, 0xf30, 0xf00,
```

```
20 WW      0xf08, 0xf0c, 0xf0f, 0xc0f, 0xa0f, 0x70f, 0x40f, 0x00f,
21 hj      0x04f, 0x09f, 0x0bf, 0x0df, 0x0ff, 0x0fa, 0x0f0, 0x0c0,
22 yd      0x090, 0xba5, 0xb86, 0x866, 0xccc, 0x999, 0x666, 0x333
23 gc      };
24 I10 #define RND RangeRand
25 jj  main ()
26 Mp  {
27 Hv3    WORD i;
28 oH     InitScreen ();
29 rM     /*** Bild zeichnen ***/
30 oy     for (i=0; i<1000; ++i)
31 Ru6       {
32 CZ        SetAPen (rp,RND(32));
33 hw        Move (rp,RND(320),12+RND(244));
34 fl        Draw (rp,RND(320),12+RND(244));
35 Z4        }
36 VA3    /*** Demo ***/
37 dR     Delay (50);
38 fh     for (i=0; i<100; i+=2)
39 1m6       ExchangeColor (&screen[0]->BitMap,0,12,&screen[1]->B
          itMap,0,12,&screen[2]->BitMap,0,12,160,244,1+RangeRan
          d(31),1+RangeRand(31));
40 gU3    Delay (50);
41 hf     for (i=0; i<31; ++i)
42 OR6       ReplaceColor (&screen[0]->BitMap,0,12,&screen[1]->Bi
          tMap,0,12,320,244,i,i+1);
43 LD3    Free ();
44 iD0 }
45 uw  InitScreen ()
46 g9  {
47 bF3    WORD i;
48 px     if (! (GfxBase = OpenLibrary ("graphics.library",0)))
49 r86       CleanUp ("Can't open graphics.library");
50 sk3    if (! (IntuitionBase = OpenLibrary ("intuition.library",0
       )))
51 oR6       CleanUp ("Can't open intuition.library");
52 xC3    for (i=2; i>=0; i--)
53 nG6       {
54 QU        if (i==0)
55 1Q9          ns.Depth = 5;
56 jW6       else
57 y99          ns.Depth = 1;  /* Hintere Screens nur eine BitPlane
             */
58 Sj6       if (! (screen[i] = OpenScreen (&ns)))
59 cz9          CleanUp ("Can't open screen");
60 yT6       }
61 Zm3    LoadRGB4 (&screen[0]->ViewPort,color,32);
62 nq     rp = &screen[0]->RastPort;
63 1W0 }
64 bG  CleanUp (str)
65 UX  char *str;
66 OT  {
67 UK3    printf ("%s\n",str);
68 kc     Free ();
69 Uk     Exit ();
70 8d0 }
71 3U  Free ()
72 6Z  {
73 1f3    WORD i;
74 kL     for (i=0; i<3; i++)
75 QM6       if (screen[i])
76 dc9          CloseScreen (screen[i]);
77 W13    if (IntuitionBase)
78 JQ6       CloseLibrary (IntuitionBase);
79 533    if (GfxBase)
80 RK6       CloseLibrary (GfxBase);
81 Jo0 }
(C) 1987 M&T
```

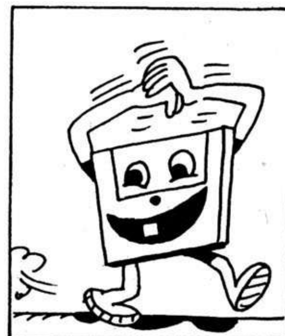
Listing 2. »demo.c« zeigt Ihnen auf einfache Weise die Verwendung der Funktionen

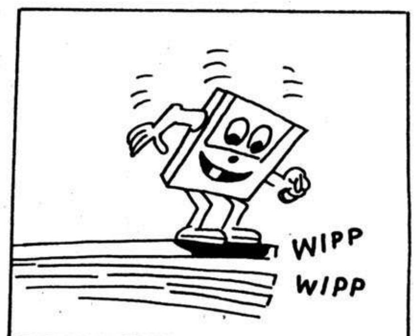
AMICUS*

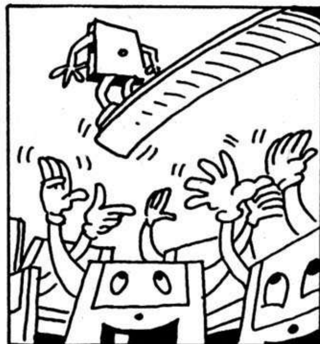
DIE ABENTEUER EINER FLOPPY-DISC

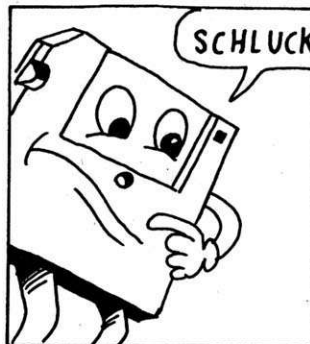


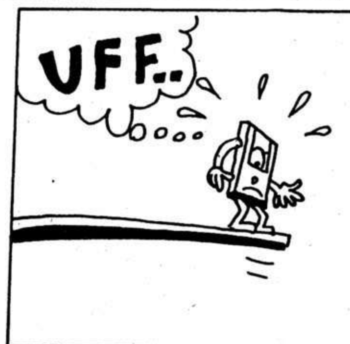


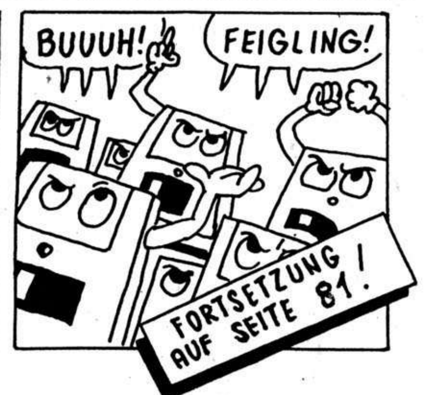












* Lat.: Der Freund

Der Weltenbauer

Haben Sie nicht auch schon davon geträumt, Planeten zu erschaffen und auf eine Umlaufbahn um eine Sonne zu bringen? Erleben Sie mit unserer Simulation »hautnah« die Massenanziehungskraft bei Himmelskörpern.

Gravitation, die Massenanziehungskraft, ist für uns moderne Menschen so selbstverständlich, daß wir uns über sie kaum noch Gedanken machen. Doch gerade deshalb fehlt bei vielen das Verständnis für dieses faszinierende physikalische Phänomen.

Mit dem Programm »DeRev« (Listing 1), welches das Experimentieren mit der Gravitation in der Welt der Planeten und Sterne erlaubt, kann man sich nicht nur das fehlende Verständnis aneignen. Der Anwender ist zudem in der Lage, ein wenig den »Schöpfer« zu spielen und neue Welten und Sonnensysteme zu erschaffen und untergehen zu lassen (Bild 1).

Das Prinzip des Programms beruht darauf, bis zu hundert Objekte auf dem Bildschirm zu verteilen und jedem seine eigene Masse und Geschwindigkeit geben zu können. Wenn man die Berechnung startet, wird die Anziehungskraft jedes einzelnen Objektes auf jedes andere berechnet. Dadurch ändern sich die Bahnen und Geschwindigkeiten, womit sehr deutlich die Wirkung starker und schwacher Gravitationskräfte zu beobachten ist.

Um ein lauffähiges Programm zu erhalten, müssen die beiden Programmteile »DeRev.MAIN.c« und »DeRev.IO.c« mit dem Checksummer abgetippt und anschließend compiliert werden. Die Compileraufrufe für den Aztec C-Compiler lauten:

```
cc DeRev.MAIN.c +l -S
cc DeRev.IO.c
```

Anschließend sind die beiden ».o«-Dateien zusammenzulinken:

```
ln -o DeRev De-
Rev.MAIN.o DeRev.IO.o -
lm32 -lc32
```

Im Anschluß daran steht das lauffähige Programm unter dem Namen »DeRev« zur Verfügung. Das Programm benötigt die »mathtrans.library«, die im aktuellen »libs«-Verzeichnis stehen muß.

Wird das Programm gestartet, erscheint zunächst nur ein dunkelblauer Bildschirm. Mit der rechten Maustaste gelangt man zum Pull-Down-Menü, mit dem das gesamte Programm gesteuert wird.

Es gibt insgesamt 19 Menüpunkte:

Start: Die Berechnung wird gestartet.

Stop: Die Berechnung wird angehalten.

Neues Objekt: Zu den schon bestehenden Objekten kann ein weiteres hinzugefügt werden. Dazu wird eine Reihe von einzelnen Requestern durchlaufen, in denen die Parameter einzustellen sind. Als erstes ist anzugeben, ob man das Objekt auf dem Bildschirm positionieren will. Wird »J« gewählt, kann mit der Maus über den Bildschirm gewandert und die gewünschte Position gewählt werden. Am oberen Bildschirmrand wird der Abstand von der Bildmitte in Millionen Kilometern angegeben. Nach Druck auf die linke Maustaste erscheint ein Fenster, in dem sich die Position noch einmal mit Hilfe von vier kreuzförmig angeordneten »Knöpfen« genauer angeben läßt. In dieses Fenster wäre man auch gekommen, hätte man sich am Anfang gegen die Positionierung auf dem Bildschirm entschieden. Je länger man auf einen Knopf drückt, um so schneller wird herauf- oder heruntergezählt. Auf die Verwendung von Proportional-Gadgets wurde verzichtet, da es kompliziert geworden wäre, mit diesen Gadgets sehr große Zahlenbereiche mit bis zu drei Stellen hinter dem Komma genau anzugeben. Ist die Position eingestellt, drückt man auf »OK«. Es erscheint ein Fenster, mit dem man die Masse des Objekts eingeben kann. Auch hier beendet »OK« die Funktion. Als letztes ist die Geschwindigkeit des Körpers einzustellen. Da sich das Objekt in beliebigen Richtungen über den Bildschirm bewegen kann, genügt es nicht, nur die Geschwindigkeit zu bestimmen. Vielmehr ist anzugeben, mit welcher Geschwindigkeit es sich in die X-Richtung (horizontal) und in die Y-Richtung (vertikal) bewegen soll. Natürlich können auch hier wie bei der Position negative Werte eingestellt werden. Ist auch diese letzte Eingabe mit »OK« abgeschlossen, erscheint das Objekt, wenn es nicht außerhalb des Bildschirms liegt, als kleiner weißer Punkt an der eingestellten Position.

Masse: Hiermit kann nachträglich, auch mitten in der Berechnung, die Masse eines Objektes geändert werden. Dazu ist zuerst das betreffende Objekt zu bestimmen. Deshalb erscheint als erstes ein Fenster, in dem diese Einstellung vorgenommen werden kann.

Position: Hier kann nachträglich die Position eines Objekts geändert werden.

Geschwindigkeit: Dient zur Änderung der Geschwindigkeit.

Löschen: Der Benutzer legt zuerst fest, welches Objekt gelöscht werden soll. Danach erscheint eine Sicherheitsabfrage. Wird diese mit »J« beantwortet, verschwindet das Objekt vom Bildschirm und wird auch nicht mehr in die Berechnung mit einbezogen. Wenn man nun zum Beispiel Objekt 2 löscht, entsteht an Stelle 2 aber kein Loch, sondern die nachfolgenden Objekte rücken in den Nummern um eins vor. Objekt 3 wird zu 2 und so weiter. Das gelöschte Objekt stellt sich nun aber an den frei gewordenen letzten Platz. Also kann man dieses Objekt wieder auf den Bildschirm bringen, indem man »Neues Objekt« anwählt und die Werte einfach nicht verändert. Nebenbei können auf diese Art und Weise auch Objekte nur für eine Weile aus der Berechnung ausgeschlossen werden.

Zeigen: Wenn man die Objekte eine Weile hat kreisen lassen, fällt es einem vielleicht schwer, unter den vielen Objekten und Bahnen noch festzustellen, welche Nummer welches Objekt hat. Deshalb kann in diesem Menüpunkt die Nummer des gesuchten Objekts angewählt werden. Drückt der Benutzer auf »OK«, wird das Objekt so lange blinken, bis die linke Maustaste gedrückt wird.

Zeitintervall: Bestimmt, wie viele Stunden zwischen zwei Berechnungsschritten vergehen sollen. Voreingestellt sind hier 24 Stunden, die ein guter Kompromiß zwischen Geschwindigkeit und Genauigkeit darstellen. Besonders bei stark exzentrischen Bahnen, also Bahnen, die stark von der



Kreisform abweichen, machen sich bei großem Zeitintervall Rechenungenauigkeiten bemerkbar. Dies liegt nicht etwa an einer schlechten Programmierung des für die Bewegung zuständigen Programmteils oder an ungenauen Mathematikroutinen des C-Compilers, sondern an dem angewandten Rechenverfahren. Dem Programmautor ist allerdings auch kein anderes Verfahren bekannt, welches hier Verwendung finden könnte. Wählt man den Zeitintervall genügend klein, werden auch sehr stark exzentrische Bahnen noch ziemlich genau berechnet.

Fixstern (JA/NEIN): Dieser Menüpunkt ist schon ein Schalter für sich. Bei jedem Anwählen verändert er sich. Aus »JA« wird »NEIN« und umgekehrt. In den meisten Fällen wird man die Bahnen von Planeten um einen Stern berechnen lassen. Da aber im allgemeinen die Massen der Planeten so gering sind, daß sie den Stern nicht sonderlich aus der Ruhelage bringen, kann man Zeit bei der Berechnung sparen, indem man »Fixstern (JA)« einstellt. Dann ist Objekt 1 unbeweglich geworden, fixiert, und es muß auf ein Objekt weniger die wirkenden Kräfte errechnet werden.

Das Programm geht davon aus, daß das Objekt 1 der Fixstern ist. Wählt man »Fixstern (NEIN)«, so wird nun auch der Stern beweglich. So kann man beispielsweise beobachten, was mit diesem und seinen Planeten geschieht, wenn ein sehr schweres Objekt vorbeifliegt.

Vergrößerung: In diesem Fenster kann festgelegt werden, wie viele Millionen Kilometer durch einen Bildpunkt repräsentiert werden. Je größer die Zahl, um so weiter entfernt man sich von den Objekten. Ihr Abstand zur Bildmitte wird kleiner, und man kann auch die äußeren Regionen eines Sonnensystems in Augenschein nehmen. Je kleiner die Zahl, um so größer werden die Bahnen dargestellt.

Uhr: Hier kann man bestimmen, ob am oberen Bildrand eine Uhr eingeblendet werden soll. Diese Uhr zeigt die verstrichenen Jahre und Tage an. Sollen neben den Jahren auch noch die Tage ausgegeben werden, hat dies den Nachteil, daß die ständige Ausgabe der Tage Zeit raubt und die Ablaufgeschwindigkeit so sinkt.

Gravo Grafik: Dieser Menüpunkt bietet die faszinierende Möglichkeit, die Stärke der Gravitation bildlich darzustellen. Der Bildschirm wird dafür von oben nach unten abgetastet. In der dabei entstehenden Grafik wird die Stärke der Gravitationsfelder der Objekte durch Höhe und Breite von »Bergen«, den sogenannten Gravitationstrichtern, dargestellt. Zuerst muß ein Dehnungsfaktor bestimmt werden, der die Höhe der Berge beeinflußt. Dann ist anzugeben, welche Rechengenauigkeit gewünscht ist. Hier sind Werte zwischen 1 und 16 erlaubt. Bei 1 wird nur jede 16. Zeile des Bildschirms in die Rechnung mit einbezogen, bei 8 jede zweite und bei 16 alle. Die Zeit, die zum Erstellen einer Grafik benötigt wird, hängt maßgeblich von der gewählten Genauigkeit, aber auch von der Anzahl der Objekte ab. Hat man »Fixstern (JA)« aktiviert, wird Objekt 1 nicht in die Berechnung mit einbezogen. Dies hat den Vorteil, daß die kleinen Gravitationstrichter der Planeten, angesichts des gigantischen Trichters der Sonne, nicht völlig verschwinden. Die Berechnung der Grafik kann durch Betätigen des Close-Gadgets beendet werden.

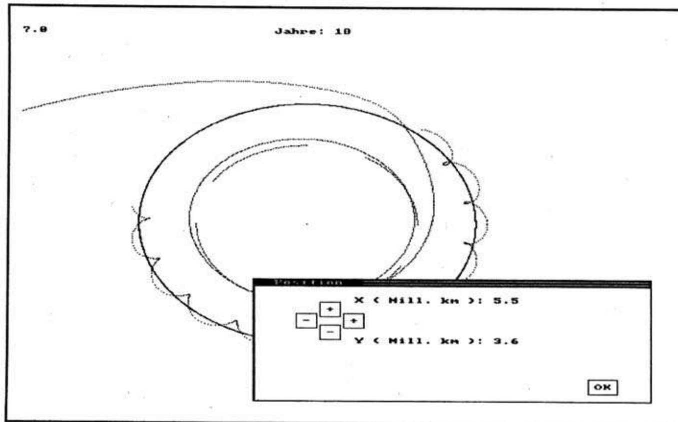


Bild 1. Ein Teil eines gedachten Sonnensystems mit den Umlaufbahnen der einzelnen Planeten

Neuer Nullpunkt: Es kann der neue Bildschirmmittelpunkt mit dem Mauszeiger bestimmt werden. Damit wird ermöglicht, Objekte, die aus dem Bildschirm hinauswandern, weiter zu verfolgen.

Bildschirm löschen: Nach einer Sicherheitsabfrage wird der Bildschirm gelöscht.

Laden: Eine zuvor gespeicherte Konstellation wird nach einer Sicherheitsabfrage geladen und kann weiter berechnet werden. Es erscheint eine Zeile, in der der Pfad- oder Dateiname anzugeben ist.

Speichern: Alle Objekte im Speicher werden auf Diskette gesichert, auch diejenigen, die zur Zeit nicht berechnet werden. Auch hier ist ein Dateiname anzugeben.

Neu: Nach einer Sicherheitsabfrage wird der Bildschirm gelöscht und die Anfangsparameter eingestellt.

Ende: Verlassen des Programms.

Tips für den Anfang

Damit man ein Gefühl für die richtigen Werte bekommt, ist unser Sonnensystem bereits voreingestellt. Das bedeutet, wenn man ein paar mal »Neues Objekt« wählt und die Werte nicht verändert, erscheinen die Körper unseres Sonnensystems auf dem Bildschirm. Wenn man das fünfmal macht, hat man vor sich die Sonne und die inneren Planeten Merkur, Venus, Erde und Mars.

Hiermit kann man besonders gut experimentieren, wenn man beispielsweise ein sehr schweres Objekt erschafft und dieses durch das innere Sonnensystem »fliegen« läßt. Beobachten Sie dabei, was mit den Planeten geschieht.

Dazu läßt man jeden Planeten mindestens einen Umlauf vollenden, damit man sieht, wie die Bahnen für gewöhnlich aussehen. Dann wählt man »Neues Objekt«. Als nächstes käme eigentlich der Jupiter. Ihn positionieren wir aber ungefähr bei X=280 und Y=-100. Damit die Wirkung deutlicher wird, erhöhen wir die Masse um das 5fache bis 30fache. Starten Sie die Berechnung und sehen Sie, was passiert...

Aber experimentieren Sie selbst. So bekommen Sie am schnellsten eine Vorstellung über die Gravitation und es macht am meisten Spaß.

(J. Brendel/D. Meier/rs)

Programmname:	DeRev
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec-C V3.4
Aufrufe:	cc DeRev. MAIN.c +I -scc DeRev.IO.c ln -o DeRevDeRev.MAIN.o DeRev.IO.o-lm32 -lc32
Bemerkung:	benötigt mathtrans.library

```
Programm : DeRev
----------------

DIO /*****************************************
nj  **   De revolutionibus orbium coelestium   **
sU  **              ( DeRev.MAIN.c )           **
4Y  **               (c) 1988 by               **
fs  **              Jürgen  Brendel            **
cI  *****************************************/
ih  #include <math.h>
W1  #include <intuition/intuition.h>
Et  #define MA   100   /* -- maximale Anzahl von Objekten -- */
JV  #define M1   101
8u  #define MENP 19    /* -- Anzahl der Menüpunkte -- */
JN  int    count,code,code2,class,Jahre=0,xx,yy,
Zc7        xalt[M1], yalt[M1], p[161];
qi0 float  gm[M1],m[M1],x[M1],y[M1],x2[M1],y2[M1],
yE7        Vx[M1],Vy[M1],Faktor,winkel,a,Vabl,xd,yd,
b5         Zeit,zeit2,gam,var2,fak,ystep,Tage=0.0;
r30 char   string[40],buffer[71],undo[71],z,t,i,anz,start,
Ij7        bflag,vflag,uflag,tflag,jflag;
dj0 extern                save(), load();
Ho  extern long           *OpenLibrary();
9q  extern struct Screen  *OpenScreen();
```

Listing 1. Mit »DeRev.MAIN.c« erstellen Sie eigene Sonnensysteme, die Wirkung der Massenanziehung lernen Sie dabei »spielend« kennen. Bitte mit dem Checksummer (Seite 159) eingeben.

```
extern struct Window  *OpenWindow();
struct Screen         *screen;
struct Window         *window, *window2;
struct IntuitionBase  *IntuitionBase;
struct GfxBase        *GfxBase;
struct RastPort       *rport, *rport2;
struct IntuiMessage   *message;
struct IntuiText       mt[MENP];
struct MenuItem        mi[MENP];
struct Menu            menu =
{
  NULL,10,0,186,10,MENUENABLED,"(c) 1988 by J. Brendel",mi
};
struct StringInfo      strinf =
{
  (UBYTE *)&buffer,(UBYTE *)&undo,0,70,0,0,0,0,0,0,0,0
};
struct Gadget          strgad =
{
  NULL,3,11,517,10,GADGHCOMP,GADGIMMEDIATE | RELVERIFY,
  STRGADGET,NULL,0,NULL,0,&strinf,NULL,0
};
struct NewScreen       hauptscreen =
{
  0,0,640,512,2,0,1,HIRES | LACE,
  CUSTOMSCREEN,NULL,NULL,NULL,NULL
};
struct NewWindow       hauptwindow =
{
  0,0,640,512,0,1, MENUPICK | MOUSEBUTTONS,
  BORDERLESS | ACTIVATE | SMART_REFRESH,
  NULL,NULL,NULL,NULL,NULL,10,10,640,512,CUSTOMSCREEN
};
struct NewWindow       parawindow =
{
  30,30,370,160,1,3,MOUSEBUTTONS,
  WINDOWDRAG | ACTIVATE,NULL,NULL,NULL,
  NULL,NULL,30,30,370,160,CUSTOMSCREEN
};
struct NewWindow       gragrawindow =
{
  30,0,520,512,1,3,CLOSEWINDOW,
  WINDOWCLOSE | WINDOWDRAG | ACTIVATE | GIMMEZEROZERO,
  NULL,NULL,(UBYTE *)" GravoGrafik  ",
  NULL,NULL,30,0,520,512,CUSTOMSCREEN
};
struct NewWindow       lswindow =
{
  0,0,521,22,1,3,GADGETUP,WINDOWDRAG | ACTIVATE,
  &strgad,NULL,NULL,NULL,NULL,0,0,520,13,CUSTOMSCREEN
};
  /* -------------------- */
 /* ---- Funktionen ---- */
/* -------------------- */
void openall()
{
  /* ---- Libraries, Screen und Window öffnen ---- */
  if(!(IntuitionBase = (struct IntuitionBase *)
    OpenLibrary("intuition.library", 0))) exit(FALSE);
  if(!(GfxBase = (struct GfxBase *)
    OpenLibrary("graphics.library", 0)))
  { CloseLibrary(IntuitionBase); exit(FALSE); }
  if(!(screen = OpenScreen(&hauptscreen)))
  { CloseLibrary(IntuitionBase); CloseLibrary(GfxBase);
    exit(FALSE); }
  hauptwindow.Screen = parawindow.Screen =
       gragrawindow.Screen = lswindow.Screen = screen;
  if(!(window = OpenWindow(&hauptwindow)))
  { CloseScreen(screen); CloseLibrary(IntuitionBase);
    CloseLibrary(GfxBase); exit(FALSE); }
  rport=window->RPort;
  /* ---- Menüs installieren ---- */
  for(t=0; t<MENP; t++)
  {
    mi[t].NextItem = (t<(MENP-1)) ? &mi[t+1] : NULL;
    mi[t].LeftEdge = 0;     mi[t].TopEdge = t*16;
    mi[t].Width    = 178;   mi[t].Height  = 10;
    mi[t].Flags    = ITEMTEXT | ITEMENABLED | HIGHCOMP;
    mi[t].MutualExclude = NULL;
    mi[t].ItemFill      = (APTR) &mt[t];
    mi[t].Command       = NULL;  mi[t].SubItem = NULL;
    mi[t].NextSelect    = NULL;
    mt[t].FrontPen      = 0;     mt[t].BackPen   = 1;
    mt[t].TopEdge       = 1;     mt[t].LeftEdge  = 5;
    mt[t].DrawMode      = JAM2;  mt[t].ITextFont = NULL;
    mt[t].NextText      = NULL;
  }
  mt[0].IText = "Start";
  mt[1].IText = "Stop";
  mt[2].IText = "Neues Objekt";
  mt[3].IText = "Masse";
  mt[4].IText = "Position";
  mt[5].IText = "Geschwindigkeit";
  mt[6].IText = "Löschen";
  mt[7].IText = "Zeigen";
  mt[8].IText = "Zeitintervall";
  mt[9].IText = "Fixstern (JA)";
  mt[10].IText = "Vergrößerung";
  mt[11].IText = "Uhr";
  mt[12].IText = "Gravo - Grafik";
  mt[13].IText = "Neuer Nullpunkt";
  mt[14].IText = "Bildschirm löschen";
  mt[15].IText = "Laden";
  mt[16].IText = "Speichern";
  mt[17].IText = "Neu";
  mt[18].IText = "Ende";
  SetMenuStrip(window, &menu);
  SetRGB4(&(screen->ViewPort), 0,  0,  0,  7);
  SetRGB4(&(screen->ViewPort), 1,  3, 15,  3);
  SetRGB4(&(screen->ViewPort), 2, 15, 15, 15);
  SetRGB4(&(screen->ViewPort), 3, 15,  0,  0);
}
void startwerte()
{
  for(t=0; t<MA; t++)
  { m[t]=1.0e6; x[t]=0.0; y[t]=0.0; Vx[t]=0.0; Vy[t]=0.0; }
  m[1]=1.989e15; x[1] =    0.0;  y[1] =  0.0;  /* Sonne */
                Vx[1] =    0.0; Vy[1] =  0.0;
  m[2]=0.333e09; x[2] =   57.9;  y[2] =  0.0;  /* Merkur */
                Vx[2] =    0.0; Vy[2] = 47.9;
  m[3]=4.870e09; x[3] =  108.2;  y[3] =  0.0;  /* Venus */
                Vx[3] =    0.0; Vy[3] = 35.0;
  m[4]=5.977e09; x[4] =  149.6;  y[4] =  0.0;  /* Erde */
                Vx[4] =    0.0; Vy[4] = 29.765;
  m[5]=0.644e09; x[5] =  227.9;  y[5] =  0.0;  /* Mars */
                Vx[5] =    0.0; Vy[5] = 24.1;
  m[6]=1.899e12; x[6] =  778.3;  y[6] =  0.0;  /* Jupiter */
                Vx[6] =    0.0; Vy[6] = 13.1;
  m[7]=5.684e11; x[7] = 1427.0;  y[7] =  0.0; /* Saturn */
                Vx[7] =    0.0; Vy[7] = 9.6;
  m[8]=8.676e10; x[8] = 2870.0;  y[8] =  0.0; /* Uranus */
                Vx[8] =    0.0; Vy[8] = 6.9;
  m[9]=1.029e11; x[9] = 4496.0;  y[9] =  0.0; /* Neptun */
                Vx[9] =    0.0; Vy[9] = 5.4;
  start=2; mt[9].IText = "Fixstern (JA)"; uflag=0;
  Faktor=1.7e9; anz=0; Zeit=86400.0; zeit2=Zeit/3600.0;
  fak=50000000.0; ystep=1.0; gam=.00667*(Zeit/1.0e5);
}
void wertaufbereiten()
{
  gm[t]=gam*m[t];
  x2[t]=(x[t]*=1.0e9);  y2[t]=(y[t]*=1.0e9);
  Vx[t]*=(Zeit*1000.0); Vy[t]*=(Zeit*1000.0);
  xalt[t]=320+(int)(x[t]/Faktor);
  yalt[t]=256+(int)(y[t]/Faktor);
}
void ifuflag()
{
  if(uflag == 1)
  {
    SetAPen(rport, 2);
    if(tflag==1)
    {
      sprintf(string,"Tage: %.1f  ",Tage);
      Move(rport, 0, 8);
      Text(rport, (int)string, (int)(strlen(string)));
    }
    sprintf(string,"Jahre: %ld  ",Jahre);
    Move(rport, 290, 8);
    Text(rport, (int)string, (int)(strlen(string)));
  }
}
void clearscreen(neum)
char neum;
{
  Move(rport, 0, 0); ClearScreen(rport); ifuflag();
  SetAPen(rport,2);
  if(neum == 1)
```

```
st4       for(t=1; t<=anz; t++)
P06         WritePixel(rport, xalt[t], yalt[t]);
6b0 }
jH  void print(s,aa,bb)
N9  char *s;  int aa,bb;
5Y  {
3I2   Move(rport2, (int)aa, (int)bb);
uC    Text(rport2, (int)s, (int)(strlen(s)));
Ch0 }
dI  void rec(s,aa,bb)
TF  char *s;  int aa,bb;
Be  {
tW2   int breite=19;
a0    if(aa == -1)
Dg    { aa=310; bb=130; breite=27; s="OK"; }
wJ    Move(rport2,(int)          aa, (int)      bb);
1P    Draw(rport2,(int)(aa+breite), (int)      bb);
tL    Draw(rport2,(int)(aa+breite), (int)(bb+19));
65    Draw(rport2,(int)          aa, (int)(bb+19));
xB    Draw(rport2,(int)          aa, (int)      bb);
E3    print(s,aa+6,bb+12);
Pu0 }
mI  void button()
Nq  {
ok2   message = (struct IntuiMessage *)GetMsg(window2->UserPor
      t);
FN    code=message->Code; class=message->Class;
gs    for(count=0; count<8000; count++);
OR    if(class == MOUSEBUTTONS)
Sv    {
LI4     if(code == SELECTUP) bflag=0;
eT      if(code == SELECTDOWN) bflag=1;
Z42   }
a50 }
yn  char booltest(wtitle,text)
gM  char *wtitle, *text;
Z2  {
742   parawindow.Title = wtitle; parawindow.Height = 110;
f6    rport2=(window2=OpenWindow(&parawindow))->RPort;
o9    vflag = 2; SetAPen(rport2, 2);
3f    rec("J",40,30); rec("N",40,60); SetAPen(rport2, 3);
pU    sprintf(string,"%s",text); print(string,88,57);
P7    while(vflag == 2)
g9    {
904     button(); vflag = 2;
sI      if(bflag == 1)
jC      {
JW6       xx=window2->MouseX; yy=window2->MouseY;
Pw        if(xx>=40 && xx<=59 && yy>=30 && yy<=49) vflag=1;
kM        if(xx>=40 && xx<=59 && yy>=60 && yy<=79) vflag=0;
rM4     }
sN2   }
df    for(count=0; count<10000; count++);
wC    CloseWindow(window2); parawindow.Height = 160;
f9    bflag = 0; return vflag;
wR0 }
HZ  float einst(kr,wtitle,text,schritt,sfak,sfak2,var,max,min,s
    max,kom,exp)
sG  char kr, *wtitle, *text, kom, exp;
Xo  float schritt,sfak,sfak2,var,max,min,smax;
wP  {
UP2   float schritt2=schritt; int fl=0; char exflag;
HL    if(kr > 0) fl=22; if(kr == 1) text=" ( km/s ): ";
JT    if(kr == 2) text=" ( Mill. km ): ";
i1    parawindow.Title = wtitle;
5W    rport2=(window2=OpenWindow(&parawindow))->RPort;
ij    SetAPen(rport2, 2); rec("",-1,0);
3N    rec("+",40+fl,30); rec("-",40+fl,60);
HJ    if(kr > 0) { rec("-",40,45); rec("+",84,45); }
H1    SetAPen(rport2, 3); exflag=0;
Hs    while(exflag == 0)
7a    {
Zx4     button(); vflag=0;
Jj      if(bflag == 1)
Ad      {
Ax6       xx=window2->MouseX; yy=window2->MouseY;
1Q        if(xx>=40+fl && xx<=59+fl && yy>=30 && yy<=49)
z08         { var+=(schritt*sfak); vflag=1; }
X26       if(xx>=40+fl && xx<=59+fl && yy>=60 && yy<=79)
588         { var-=(schritt*sfak); vflag=1; }
WJ6       if(xx>=310 && xx<=337 && yy>=130 && yy<=159)
jk8         exflag=1;
Dj6       if(kr > 0)
Jm        {
G08         if(xx>=40 && xx<=59 && yy>=46 && yy<=65)
CnA           { var2-=(schritt*sfak); vflag=1; }
Ar8         if(xx>=84 && xx<=103 && yy>=46 && yy<=65)
8hA           { var2+=(schritt*sfak); vflag=1; }
Sx6       }
Ty4     }
s3      if(var > max) { var = min;  schritt = schritt2; }
rC      if(var < min) { var = max;  schritt = schritt2; }
7T      if(var2 > max) { var2 = min;  schritt = schritt2; }
wG      if(var2 < min) { var2 = max;  schritt = schritt2; }
VS      if(exp == 1 && schritt == schritt2)
Vy      {
1S6       schritt=1.0;
1g        while( (schritt*=10.0) < var ); schritt/=1.0e4;
c74     }
7o      if(kom > 0 && exp == 0)
a3      {
KU6       if(kom == 2)
vu8         sprintf(string,"%s%.3f        ",text,var/sfak);
WJ6       else
e7        {
5U8         if(kr == 0)
BBA           sprintf(string,"%s%.1f         ",text,var/sfak);
DY8         if(kr == 1)
yXA           sprintf(string,"Y%s%.3f       ",text,var/sfak);
Lc8         if(kr == 2)
uRA           sprintf(string,"Y%s%.1f       ",text,var/sfak);
pK6       }
qL4     }
gT      else
HG6       sprintf(string,"%s%.0f          ",text,var/sfak);
OW4     if(exp == 1)
MN6       sprintf(string,"%s%.3e   ",text,var/sfak);
Mr4     if(kr == 0) print(string,88,57);
1Y      else
tM      {
uu6       print(string,95,83);
Rm        if(kr == 1)
Vm8         sprintf(string,"X%s%.3f       ",text,var2/sfak);
Zq6       if(kr == 2)
Rg8         sprintf(string,"X%s%.1f       ",text,var2/sfak);
YQ6       print(string,95,30);
4Z4     }
nX      if(vflag == 1)
2V      {
SZ6       schritt*=sfak2;
gJ        if(schritt>smax) schritt=smax;
2m        if(( kr>0 && (xx>=40+fl && xx<=59+fl) ) || kr == 0
          )
6Z        {
tt8         if(schritt*sfak > abs(var)/5.0 && yy>=60)
wzA           schritt=abs(var)/5.0/sfak;
D16       }
3q        else
yz8         if(schritt*sfak > abs(var2)/5.0 && xx<=40+fl)
rAA           schritt=abs(var2)/5.0/sfak;
Hm4     }
7u      else
Fi      {
Bt6       schritt=schritt2;
Sy        if(exp == 1)
I1        {
oF8         schritt=1.0;
oT          while( (schritt*=10.0) < var ); schritt/=1.0e4;
Pu6       }
Qv4     }
Rw2   }
qQ    CloseWindow(window2); bflag=0; return var;
Ty0 }
EI  void lsgadwart()
Ru  {
GH2   window2 = OpenWindow(&lswindow);
Tv    class = GADGETDOWN;
59    while(class != GADGETUP)
Vy    {
ws4     message = (struct IntuiMessage *)GetMsg(window2->UserP
        ort);
vL      class = message->Class;
c72   }
kY    strcpy(undo,buffer); CloseWindow(window2);
e90 }
```

Listing 1. (Fortsetzung)

```
void posinp(ob)
int ob;
{
  char naeheflag=1; int bp;
  bp = (int)(booltest(" Achtung! ",
              "Bildschirmposition? (J/N)"));
  for(count=1; count<10000; count++);
  while(naeheflag == 1)
  {
    if(bp == 1)
    {
      code = 0; class = 0; SetAPen(rport, 2);
      while(!(class == MOUSEBUTTONS && code == SELECTUP))
      {
        message = (struct IntuiMessage *)GetMsg(window->Us
        erPort);
        class=message->Class; code=message->Code;
        xd=(float)((window->MouseX)-320);
        yd=(float)((window->MouseY)-256);
        sprintf(string," X ( Millionen km ): %.1f        ",

                xd*Faktor/1.0e9);
        Move(rport, 0, 8);
        Text(rport, (int)string, (int)(strlen(string)));
        sprintf(string,"  Y ( Millionen km ): %.1f        "
        ,
                yd*Faktor/-1.0e9);
        Move(rport, 290, 8);
        Text(rport, (int)string, (int)(strlen(string)));
      }
      SetAPen(rport,0); RectFill(rport, 0, 0, 640, 9);
      ifuflag();
    }
    else
    { xd=x[ob]/Faktor; yd=y[ob]/Faktor; }
    var2=xd*Faktor/1.0e9;
    y[ob] = 1.0e9*einst(2," Position  ","",0.1,1.0,
     1.1,yd*Faktor/-1.0e9,1.0e6,-1.0e6,100000.0,1,0);
    x[ob]=var2*1.0e9; y[ob]=-y[ob]; naeheflag=0;
    for(t=1; t<anz; t++)
      if(abs(x[ob]-x[t]) < 100000000.0 && t!=ob &&
        abs(y[ob]-y[t]) < 100000000.0) naeheflag = 1;
  }
  SetAPen(rport, 01);
  WritePixel(rport, xalt[ob], yalt[ob]);
}
void tausch(xt,yt)
int xt,yt;
{
  x[xt]=x[yt]; y[xt]=y[yt];
  x2[xt]=x2[yt]; y2[xt]=y2[yt];
  xalt[xt]=xalt[yt]; yalt[xt]=yalt[yt];
  Vx[xt]=Vx[yt]; Vy[xt]=Vy[yt];
  m[xt]=m[yt]; gm[xt]=gm[yt];
}
  /* --------------------- */
 /* --- Menü - Punkte --- */
/* --------------------- */
void bewegung()
{
  register char tt,ii;
  while(class != MENUPICK)
  {
    message = (struct IntuiMessage *)GetMsg(window->UserPo
    rt);
    class = message->Class;
    if(uflag == 1)
    {
      SetAPen(rport, 2); Tage+=zeit2/24.0;
      while(Tage>365.0) { Tage-=365.0; Jahre++; jflag=1; }
      if(Jahre>2000000000) Jahre=0;
      if(zeit2 < 8760.0 && tflag==1)
      {
        sprintf(string,"Tage: %.1f  ",Tage);
        Move(rport, 0, 8);
        Text(rport, (int)string, (int)(strlen(string)));
      }
      if(jflag == 1)
      {
        sprintf(string,"Jahre: %ld  ",Jahre);
        Move(rport, 290, 8); jflag=0;
        Text(rport, (int)string, (int)(strlen(string)));
      }
    }
    for(tt=start; tt<=anz; tt++)
    { x2[tt]=x[tt]; y2[tt]=y[tt]; }
    for(tt=start; tt<=anz; tt++)
    {
      SetAPen(rport,3); WritePixel(rport,xalt[tt],yalt[tt])
      ;
      xalt[tt]=320+(int)(x[tt]/Faktor);
      yalt[tt]=256+(int)(y[tt]/Faktor);
      SetAPen(rport,2); WritePixel(rport,xalt[tt],yalt[tt])
      ;
      y[tt]-=Vy[tt]; x[tt]-=Vx[tt];
      for(ii=1; ii<=anz; ii++)
      {
        if(ii!=tt)
        {
          xd=x[tt]-x2[ii]; yd=y[tt]-y2[ii]+1.0e-15;
          winkel=atan(yd/xd); xd/=1.0e6; yd/=1.0e6;
          Vabl = (xd>0.0) ? (gm[ii]/(xd*xd+yd*yd)) :
                           (-gm[ii]/(xd*xd+yd*yd));
          Vx[tt]+=cos(winkel)*Vabl*Zeit;
          Vy[tt]+=sin(winkel)*Vabl*Zeit;
        }
      }
    }
  }
}
void neu()
{
  float ber=1; posinp((int)(++anz));
  while( (ber*=10.0) < m[anz] ); ber/=1.0e10;
  m[anz] = einst(0," Masse  ","Trillionen Tonnen: ",
       ber,1.0,1.15,m[anz]/1.0e6,1.0e12,0.00001,1.0e11,0,1
       );
  m[anz]*=1.0e6; gm[anz]=gam*m[anz];
  Vy[anz]/=Zeit; var2=Vx[anz]/Zeit/-1000.0;
  Vy[anz] = einst(1," Geschwindigkeit  ","",0.001,
     1.0,1.15,Vy[anz]/1000.0,1.0e4,-1.0e4,1000.0,1,0);
  Vx[anz]=var2*Zeit*-1000.0; Vy[anz]*=1000.0*Zeit;
  x2[anz]=x[anz]; y2[anz]=y[anz];
  xalt[anz]=320+(int)(x[anz]/Faktor);
  yalt[anz]=256+(int)(y[anz]/Faktor);
  SetAPen(rport, 2); WritePixel(rport,xalt[anz],yalt[anz]);
}
void masse()
{
  float ber=1;
  int obn = (int)(einst(0," Masse  ","Objekt: ",
     1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0,0));
  while( (ber*=10.0) < m[obn] ); ber/=1.0e10;
  m[obn] = einst(0," Masse  ","Trillionen Tonnen: ",
       ber,1.0,1.15,m[obn]/1.0e6,1.0e12,0.00001,1.0e11,0,1)
       ;
  m[obn]*=1.0e6; gm[obn]=gam*m[obn];
}
void position()
{
  int obn = (int)(einst(0," Position  ","Objekt: ",
     1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0,0));
  posinp(obn); SetAPen(rport,2); x2[obn]=x[obn]; y2[obn]=y[
  obn];
  WritePixel(rport, (xalt[obn]=320+(int)(x[obn]/Faktor)),
    (yalt[obn]=256+(int)(y[obn]/Faktor)));
}
void geschw()
{
  int obn = (int)(einst(0," Geschwindigkeit  ","Objekt: ",
     1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0,0));
  Vy[obn]/=Zeit; var2=Vx[obn]/Zeit/-1000.0;
  Vy[obn] = einst(1," Geschwindigkeit  ","",0.001,
     1.0,1.15,Vy[obn]/1000.0,1.0e4,-1.0e4,1000.0,1,0);
  Vx[obn]=var2*Zeit*-1000.0; Vy[obn]*=1000.0*Zeit;
}
void loesch()
{
  int obn = (int)(einst(0," Löschen  ","Objekt: ",
     1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0,0));
  if(((int)(booltest(" Achtung! ",
                 "Wirklich löschen? (J/N)"))) == 1)
  {
    if(obn==1) { start=1; mt[9].IText="Fixstern (NEIN)"; }
    tausch(M1,obn); SetAPen(rport,0);
    WritePixel(rport,xalt[obn],yalt[obn]);
    if(obn < anz--)
    {
```

```
046         for(t=obn; t<=anz; t++)
vA8           tausch(t,t+1);
Vw6         tausch(anz+1,M1);
Hm4       }
In2     }
Jo0 }
rJ  void zeig()
Hk  {
ts2   int obn = (int)(einst(0," Zeigen  ","Objekt: ",
2E5      1.0,1.0,1.0,1.0,(float)anz,1.0,1.0,0,0)); t=0;
HJ2   for(count=0; count<5500; count++)
Lo    {
sa4     SetAPen(rport, (int)t); t++; if(t>3) t=0;
6u      WritePixel(rport, xalt[obn], yalt[obn]);
Sx2   }
pt    SetAPen(rport,2); WritePixel(rport,xalt[obn],yalt[obn]);
Uz0 }
5k  void zeit()
Sv  {
q02   for(t=1; t<MA; t++) { Vx[t]/=Zeit; Vy[t]/=Zeit; }
Pz    Zeit = einst(0," Zeit  ","Zeit (h): ",0.1,3600.0,
R46      1.1,Zeit,9.0e7,0.09,1000.0,1,0);
Pn2   for(t=1; t<MA; t++) { Vx[t]*=Zeit; Vy[t]*=Zeit; }
0j    gam=.00667*(Zeit/1.0e5); zeit2=Zeit/3600.0;
zz    for(t=1; t<MA; t++) gm[t]=gam*m[t];
d80 }
C5  void vergr()
b4  {
Fu2   Faktor = einst(0," Vergrößerung  ","Mill. km pro Punkt:
      ",0.1,
qD7        1.0,1.1,Faktor/1.0e9,1000.0,0.0,100.0,1,0);
z42   Faktor*=1.0e9; SetAPen(rport, 3);
Z1    for(t=1; t<MA; t++)
pL    { xalt[t]=320+(int)(x[t]/Faktor);
qQ4     yalt[t]=256+(int)(y[t]/Faktor); }
mH0 }
cx  void uhr()
kD  {
eH2   if(((int)(booltest(" Uhr ","Uhr? (J/N)"))) == 1)
mF    {
7R4     uflag=1; tflag=0;
s8      if(((int)(booltest(" Uhr ",
23B             "Auf null setzen? (J/N)"))) == 1)
hw4     { Tage=0.0; Jahre=0; }
m5      if(((int)(booltest(" Uhr ","Tage? (J/N)"))) == 1)
Yu6       tflag=1;
Pf4     SetAPen(rport,0); RectFill(rport,0,0,640,9);
nk      SetAPen(rport,2);
6o      if(tflag==1)
wP      {
Ec6       sprintf(string,"Tage: %.1f  ",Tage);
tH        Move(rport, 0, 8);
co        Text(rport, (int)string, (int)(strlen(string)));
4Z4     }
hu      sprintf(string,"Jahre: %ld            ",Jahre);
Qh      Move(rport, 290, 8); jflag=0;
gs      Text(rport, (int)string, (int)(strlen(string)));
8d2   }
yl    else
6Z    {
R04     SetAPen(rport, 0); RectFill(rport, 0, 0, 640, 9);
n9      uflag=0;
D12   }
Ej0 }
sd  void gragraf()
Cf  {
d02   register char ii; float y0,x0,v,xl,yl;
ns    register int yyy,xt,yt=425,xstart=96;
nP    fak = einst(0," Vergrößerung  ","Dehnungsfaktor: ",1.0,
JW5      1.0,1.5,fak,500000000.0,1.0,10000000.0,0,0);
gV2   ystep = 16.0/einst(0," Genauigkeit  ",
Z03    "Schritte pro Zeile: ",1.0,1.0,1.0,ystep,16.0,1.0,1.0,0,
       0);
5v2   if(window2=OpenWindow(&gragrawindow))
Kn    {
w64     rport2=window2->RPort;
KH      for(yyy=1; yyy<=512; yyy+=16)
Nq      {
5y6       if((struct IntuiMessage *)GetMsg(window2->UserPort))
N08         goto gragraschluss;
Yd6       xl=(float)(2*xstart); yt+=2; yl=(float)yt;
hQ        for(y0=(float)yyy; y0<=(float)yyy+16.0; y0+=ystep)
Sv        {
I48         xt=1;
kJ          for(x0=1.0; x0<=640.0; x0+=4.0)
Vy          {
L5A           v=0.0;
N2            for(ii=start; ii<=anz; ii++)
Y1            {
sbC             if(v < 5000.0)
a3              {
F1E               xd=(x0-(float)320)*Faktor-x[ii]+1.0e-18;
Ej                yd=(y0-(float)256)*Faktor-y[ii];
3j                winkel=atan(yd/xd); xd/=1.0e6; yd/=1.0e6;
0G                v+=((gm[ii]/(xd*xd+yd*yd))*fak/1.0e4);
jEC             }
kFA           }
2C            if(v > 800.0) v=800.0;
yr            p[xt]=(p[xt++]+(int)v)/2;
nI8         }
oJ6       }
Q1        for(xt=1, x0=1.0; x0<=640.0; x0+=4.0)
mF        {
WD8         SetAPen(rport2, 0);
Sb          Move(rport2, (int)(xl+1.0), yt+1);
k0          Draw(rport2, (int)(xl+1.0), yt-p[xt]+1);
0R          Move(rport2, (int)xl, (int)yl);
Xb          xl=2.0*((float)xstart+x0/4.0); yl=(float)(yt-p[xt])
            ;
DI          SetAPen(rport2, 3); Draw(rport2,(int)xl,(int)yl);
zR          p[xt++]=0;
yT6       }
L8        Draw(rport2, (int)xl, yt); xstart-=3;
0V4     }
gg      Wait(1 << window2->UserPort->mp_SigBit);
xW0 gragraschluss:
r84     CloseWindow(window2);
4Z2   }
23    ystep=16.0/ystep;
6b0 }
8J  void neunull()
4X  {
rf2   class = 0; SetAPen(rport, 2);
q9    while(!(class == MOUSEBUTTONS && code == SELECTUP))
7a    {
ML4     message = (struct IntuiMessage *)GetMsg(window->UserPo
        rt);
pZ      class=message->Class; code=message->Code;
uR      xx=(window->MouseX)-320; yy=(window->MouseY)-256;
8C      sprintf(string," X: %ld   ",xx);
7V      Move(rport, 0, 8);
q2      Text(rport, (int)string, (int)(strlen(string)));
oy      sprintf(string,"  Y: %ld    ",yy);
Dp      Move(rport, 290, 8);
t5      Text(rport, (int)string, (int)(strlen(string)));
Lq2   }
C0    for(t=1; t<MA; t++)
9h    { x2[t]=(x[t]-=(float)xx*Faktor);
gP4     y2[t]=(y[t]-=(float)yy*Faktor);
bX      xalt[t]-=xx; yalt[t]-=yy; }
gK2   clearscreen(1);
Rw0 }
vw  void speichern()
Ps  {
Df2   lswindow.Title = "  Save   "; lsgadwart(); save();
V00 }
61  void laden()
Tw  {
XV2   if((booltest(" Achtung! ",
Im5     "Wirklich laden? (J/N)")) == 1)
Wz2   {
cL4     lswindow.Title = "  Load  ";
CG      lsgadwart(); SetAPen(rport, 0);
RS      for(t=1; t<=anz; t++)
yZ6       WritePixel(rport, xalt[t], yalt[t]);
T14     load(); SetAPen(rport, 2);
sz      if(start == 2)
B06       mt[9].IText = "Fixstern (JA)";
XK4     else
ma6       mt[9].IText = "Fixstern (NEIN)";
YZ4     for(t=1; t<=anz; t++)
jo6       WritePixel(rport, (xalt[t]=320+(int)(x[t]/Faktor)),
558         (yalt[t]=256+(int)(y[t]/Faktor)));
nI2   }
oJ0 }
```

Listing 1. (Fortsetzung)

```
9X2   /* ---------------------- */
yt1  /* ---- Hauptprogramm ---- */
BZ0 /* ---------------------- */
Zv  main()
pI  {
LM2   openall(); startwerte(); clearscreen(0); code=0; class=0;
st    for(t=1; t<MA; t++) wertaufbereiten();
PT    while(1)
tM    {
BX4     while(class != MENUPICK)
9E      { message = (struct IntuiMessage *)GetMsg(window->User
        Port);
v26       code = message->Code; class=message->Class; }
xi4     if (code == MENUNULL) { code = 0; class = 0; }
re      else
zS      {
8S6       class = 0;
04        switch( ITEMNUM (code) )
2V        {
vT8         case 0:                       /* Start */
YHA           code2 = code; SetAPen(rport, 2);
xy            for(t=1; t<=anz; t++)
U5C             WritePixel(rport, xalt[t], yalt[t]);
B3A           bewegung();
QQ            if((code = message->Code)==MENUNULL)
C2C             code = code2;
9IA           break;
uT8         case 1:                       /* Stop */
G8A           code = 0; break;
ey8         case 2:                       /* Neues Objekt */
7CA           code = 0; neu(); break;
1w8         case 3:                       /* Masse */
a1A           code = 0; if(anz>0) masse(); break;
VZ8         case 4:                       /* Position */
zUA           code = 0; if(anz>0) position(); break;
AA8         case 5:                       /* Geschwindigkeit */
rfA           code = 0; if(anz>0) geschw(); break;
my8         case 6:                       /* Löschen */
x7A           code = 0; if(anz>0) loesch(); break;
7I8         case 7:                       /* Zeigen */
ORA           code = 0; if(anz>0) zeig(); break;
Ev8         case 8:                       /* Zeitintervall */
JmA           code = 0; zeit(); break;
HL8         case 9:                       /* Fixstern (JA/NEIN) *
            /
bKA           code = 0;
fl            if(start == 1)
rB            { start=2; mt[9].IText = "Fixstern (JA)"; }
OB            else
M9            { start=1; mt[9].IText = "Fixstern (NEIN)"; }
Wf            break;
j48         case 10:                       /* Vergrößerung */
meA           code = 0; vergr(); clearscreen(1); break;
yr8         case 11:                      /* Uhr */
frA           code = 0; uhr(); break;
zo8         case 12:                      /* Gravo - Grafik */
I3A           code = 0; gragraf(); break;
D18         case 13:                      /* Neuer Nullpunkt */
itA           code = 0; neunull(); break;
aK8         case 14:                      /* Bildschirm löschen *
            /
qZA           code = 0;
1j            if((booltest(" Achtung! ",
GFN                       "Bildschirm löschen ? (J/N)")) == 1)
4iC             clearscreen(1);
ktA           break;
Lc8         case 15:                      /* Laden */
hHA           code = 0; laden(); break;
XK8         case 16:                      /* Speichern */
1NA           code = 0; if(anz>0) speichern(); break;
E68         case 17:                      /* Neu */
0jA           code = 0;
vt            if((booltest(" Achtung! ",
hiN                       "Wirklich neu? (J/N)")) == 1)
3vA           { startwerte();
wxC             for(t=1; t<MA; t++) wertaufbereiten();
ad              clearscreen(); }
w5A           break;
GQ8         case 18:                      /* Ende */
8rA           code = 0;
31            if((booltest(" Achtung! ",
J2N                       "Wirklich beenden? (J/N)")) == 1)
2VA           {
2VC             ClearMenuStrip(window); CloseWindow(window);
C3              CloseScreen(screen); CloseLibrary(IntuitionBase
                );
s8              CloseLibrary(GfxBase); exit(TRUE);
AfA           }
Bg6         }
Ch4       }
Di2     }
Ej0 }
(C) 1987 M&T
```

Listing 1. (Schluß)

Programmname:	DeRev.IO.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztek C V3.4
Aufrufe:	cc DeRev.IO.c
Bemerkung:	Dieses Programm bitte zuerst compilieren, danach »DeRev.Main.c«

```
Programm : DeRev
----------------

RW0 /*********************************************
C0  **   De revolutionibus orbium coelestium  I/O   **
DF  **               ( DeRev.IO.c )                 **
Dq  **                (c) 1988  by                  **
Ju  **               Jürgen Brendel                 **
qW  *********************************************/
HQ  #define MA  100
AE  extern char   t,anz,start,buffer[71];
wK  extern float  x[MA],y[MA],x2[MA],y2[MA],Vx[MA],Vy[MA],
1iE               m[MA],gm[MA],Zeit,gam;
in0 extern long   xalt[MA],yalt[MA];
zR  void save()
9c  {
012   int handle,nix;
mz    if((handle = creat(buffer, nix)) != -1)
Cf    {
qs4     write(handle,  &anz, sizeof(char));
ff      write(handle,&start, sizeof(char));
YD      write(handle,     x, sizeof(x));
jQ      write(handle,     y, sizeof(y));
Ms      write(handle,    x2, sizeof(x2));
W4      write(handle,    y2, sizeof(y2));
su      write(handle,    Vx, sizeof(Vx));
43      write(handle,    Vy, sizeof(Vy));
sB      write(handle,     m, sizeof(m));
rm      write(handle, &Zeit, sizeof(float));
6h      close(handle);
Sx2   }
Ty0 }
IV  void load()
Ru  {
gJ2   int handle,nix;
3t    if((handle = open(buffer, 01, nix)) != -1)
Ux    {
VR4     read(handle,  &anz, sizeof(char));
xG      read(handle,&start, sizeof(char));
q9      read(handle,     x, sizeof(x));
zK      read(handle,     y, sizeof(y));
4x      read(handle,    x2, sizeof(x2));
C7      read(handle,    y2, sizeof(y2));
QT      read(handle,    Vx, sizeof(Vx));
af      read(handle,    Vy, sizeof(Vy));
WT      read(handle,     m, sizeof(m));
mS      read(handle, &Zeit, sizeof(float));
UY      close(handle); gam=.00667*(Zeit/1.0e5);
77      for(t=1; t<MA; t++) gm[t]=gam*m[t];
1G2   }
mH0 }
(C) 1987 M&T
```

Listing 2. »DeRev.IO.c« ist als erstes Listing zu compilieren. Das Hauptprogramm benötigt diese Routine. Bitte mit dem Checksummer (Seite 159) eingeben.

»Floppy-Speeder« mit Niveau

Haben Sie sich auch schon geärgert, wenn der Amiga beim Laden von Directories und Programmen von Diskette wie wild auf dieser »schrubbt«? Dies ist ein Zeichen dafür, daß die Filestruktur auf der Diskette mächtig durcheinander ist. »FastLoadCopy« beseitigt diesen Umstand und räumt auf Ihren Disketten ordentlich auf. Die Geschwindigkeit beim Laden wird dadurch beträchtlich erhöht. So ganz nebenbei ist unser Listing auch noch ein schnelles Kopierprogramm.

Der Amiga ist ein schneller Computer — solange er keine Directories von Disketten laden soll. Durch die komplizierte Organisation des Inhaltsverzeichnisses muß der Lesekopf des Disketten-Laufwerkes oft bewegt werden, was als »Sägen« nur zu gut hörbar ist. »FastLoadCopy« schafft Abhilfe und sortiert die Blöcke des Directory und aller Programme. Dadurch werden die Diskettenzugriffe wesentlich beschleunigt und das »Sägen« reduziert. Einzige Voraussetzung zum Arrangieren der Disketten ist 1 MByte Hauptspeicher.

Für die Lesegeschwindigkeit von Amiga-Diskettenlaufwerken gibt es eine einfache Gleichung: Je mehr sich der Schreib-/Lesekopf bewegt, desto niedriger ist die Lesegeschwindigkeit. Beim Laden eines Directory (oder dem Doppelklick auf ein Diskettensymbol) bekommt man das besonders stark zu spüren: Der Schreib-/Lesekopf bewegt sich hin und her, daß es eine wahre Freude ist. Entsprechend lang dauert es, bis der Computer das Inhaltsverzeichnis eingelesen hat. Um das Ganze zu beschleunigen, müßte man also die Anzahl der Bewegungen reduzieren — genau das macht »FastLoadCopy«.

Kopie schlägt Original!

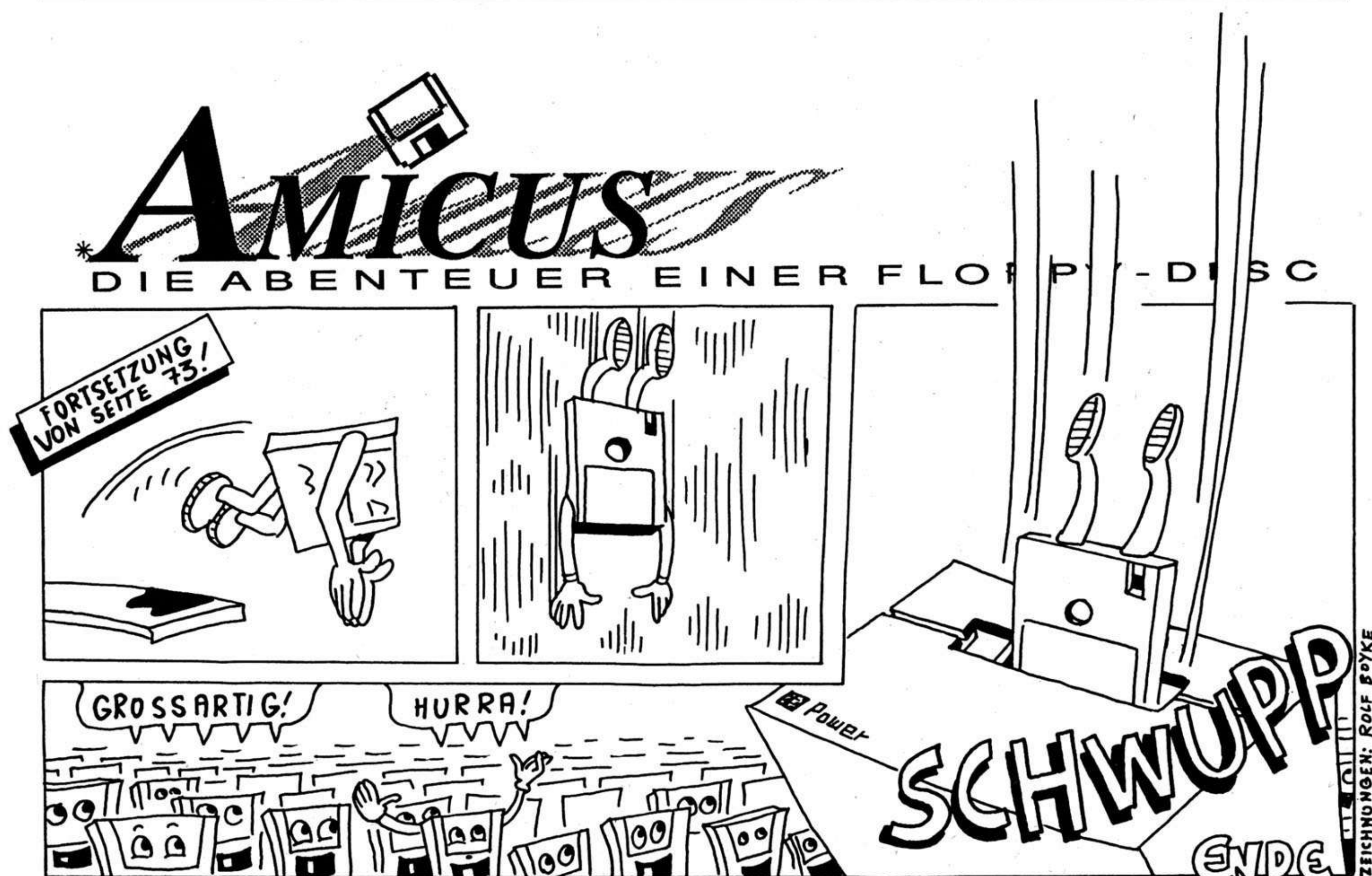
Dieses Programm funktioniert nach einem einfachen Prinzip: Zuerst einmal werden alle Blöcke der zu bearbeitenden Diskette in den Speicher gelesen. Daher ist es auch unbedingt nötig, daß Ihr Amiga mindestens 1 MByte Speicher besitzt — auf eine Amiga-Diskette passen ja 880 KByte. Stehen die Daten im Speicher, werden sie vom Programm nach einem bestimmten Schema sortiert. Dadurch wird die optimale Geschwindigkeit beim Lesen des Directory und Laden von Programmen erreicht. Nach dem Sortiervorgang — er dauert etwa eine halbe Minute — werden die nun sortierten Blöcke wieder auf die Diskette zurückgeschrieben.

Vielleicht werden Sie nun fragen: »Wieso heißt das Programm dann FastLoadCopy und nicht einfach FastLoad?«. Nun, nachdem eine Diskette beschrieben worden ist, kann man wahlweise eine weitere Diskette mit denselben Daten (die ja noch im Speicher stehen) bespielen. Unser Listing läßt sich also auch als Kopierprogramm, das zudem noch schnell ist, verwenden.

Eingabehinweise

Tippen Sie bitte Listing 1 mit dem Checksummer (Seite 159) ab. »FastLoadCopy« ist in Assembler geschrieben. Damit auch Leser, die keinen Assembler besitzen, das Programm eingeben können, haben wir dieses als DATA-Lader abgedruckt. Dieser erzeugt das Programm »FLC« auf Diskette. Den Quellcode finden Sie in Listing 2.

Wenn Sie in Ihrem Computer nur 1 MByte Speicher haben,



beachten Sie bitte: Zusätzliche Diskettenlaufwerke und Festplatten sollten vor dem Start des Programmes entfernt werden, da diese zu viel Speicherplatz verbrauchen. Um das Programm zu starten, tippen Sie im CLI:

```
FLC
```

Legen Sie nun die Diskette, die Sie bearbeiten wollen, in das interne Diskettenlaufwerk des Amiga und drücken Sie die linke Maustaste. Sollten Sie nun die Meldung »Zu wenig Speicher« erhalten, so beachten Sie bitte die Hinweise in Tabelle 1. Ist genügend Speicher vorhanden, so lädt das Programm die Daten von der Diskette in den Computer (»...loading«). Ist das Laden beendet, so werden die Blöcke sortiert (»...changing«).

Dabei treten bei fast allen Disketten einige Fehlermeldungen auf, zum Beispiel »Fehler in Block $032b. Disk u.U. fehlerhaft! Fehlernummer 1«. Wichtig ist in diesem Fall die Fehlernummer (siehe auch Tabelle 2). Bei der abgedruckten Version werden Fehlermeldungen mit Text ausgegeben. Auf der Diskette zu diesem Sonderheft befindet sich noch eine Version, welche kurze Kommentare ausgibt (sie trägt den Namen »FLCO«).

Trägt die Fehlermeldung die Nummer »1«, so hat das für den Ablauf des Programmes keine Bedeutung. Sie können in diesem Fall also getrost fortfahren. Höhere Fehlernummern als »1« müssen nicht unbedingt problematisch sein, man sollte die Daten aber vorsichtshalber erst einmal auf eine Versuchsdiskette speichern. Bei Fehlernummer »8« wird der Sortiervorgang abgebrochen, da es sich hierbei um einen Fehler handelt, der es unmöglich macht, das Programm fortzusetzen.

Nummer	Kurzbeschreibung des Fehlers
1	Fehlendes "First Data" in FileListBlock
2	Falscher "Parent" - Eintrag
3	Datenblöcke nicht zusammenhängend
4	Falscher "Parent" in FileListBlock
5	Falsches "Extension" in FileListBlock
6	Erstblockeintrag fehlt in FileHeader
	Falscher "HeaderKey" im Block
8	HardError! Keine Wiederaufnahme des Sortierens möglich: beispielsweise kein DOS-Format, falsche Bitmap, Programmfehler

Fehlernummer 1 ist unbedenklich. Bei Fehlernummern größer als 1, sollte zuerst auf eine Versuchsdiskette gespeichert werden.

Bei Fehlernummer 8 bricht das Programm selbständig ab.

Tabelle 2. Die Fehlermeldungen und ihre Bedeutung

Sortieren ist alles

Sind die Daten schließlich sortiert, so werden Sie aufgefordert, die Zieldiskette einzulegen und die linke Maustaste zu drücken. Normalerweise sind Quell- und Zieldiskette natürlich identisch, da die Blöcke ja sortiert wieder auf die Quelldiskette zurückgeschrieben werden sollen. Sie können aber natürlich Ihre Daten genausogut auf eine andere Diskette speichern und FastLoadCopy so als Kopierprogramm verwenden. Haben Sie beim Sortieren höhere Fehlernummern als 1 bekommen, verwenden Sie bitte ebenfalls eine andere Diskette als Zieldiskette, um Datenverluste zu vermeiden! Die Zieldiskette darf natürlich nicht schreibgeschützt sein.

Nachdem die Daten auf die Zieldiskette geschrieben wurden (»...saving«), haben Sie die Wahl, mit Hilfe der linken Maustaste das Programm zu verlassen oder es mit der rechten Maustaste wieder zu starten. Haben Sie die rechte Maustaste gedrückt, so folgt die Frage

```
LMaustaste = Diskkopie
RMaustaste = Ramkopie
```

»Diskkopie« bedeutet, daß das Programm wieder von vorne abläuft; mit »Ramkopie« läßt sich eine weitere Diskette mit den Daten beschreiben, die noch im Speicher sind. Sie werden wiederum aufgefordert, die Zieldiskette einzulegen und die linke Maustaste zu drücken. FastLoadCopy ist also auch als Mehrfach-Kopierprogramm gut geeignet. Interessierte finden in Tabelle 3 eine kurze Beschreibung der Funktionsweise des Programms. *(Paul Sponagl/ A. Lietz/M. Jobst/rs)*

Bei Computern mit 1 MByte Speicher

Da eine Diskette schon 880 K im Speicher belegt, ist nur noch sehr wenig freier Platz im Speicher — zu wenig, um zum Beispiel die Workbench, eine RAM-Disk oder den Speicherbereich aufzunehmen, der für externe Diskettenlaufwerke und Festplatten erforderlich ist. Auch die Benutzung von anderen Programmen vor dem Start von »FastLoadCopy« verbraucht meist zu viel kostbaren Speicherplatz.

Am besten kopieren Sie das Programm FastLoadCopy nach dem Abtippen in das Hauptverzeichnis oder das Verzeichnis »c« Ihrer Workbenchdiskette. Entfernen Sie sämtliche externen Laufwerke und lösen Sie mit <CTRL COMMODORE AMIGA> einen Reset aus.

Legen Sie Ihre Workbenchdiskette in das interne Laufwerk und warten Sie, bis die Einschaltmeldung auf dem Bildschirm erscheint. Dann drücken Sie sofort <CTRL D>, um die »startup-sequence« abzubrechen, die Ihr Computer nun normalerweise ausführen würde. Jetzt starten Sie FastLoadCopy. Nun sollten eigentlich keine Probleme mehr auftreten.

Bei Computern mit mehr als 1 MByte Speicher

Hier kann es eigentlich nur passieren, daß ein bereits geladenes Programm zuviel Speicherplatz verbraucht. Schalten Sie also laufende Programme ab oder benutzen Sie FastLoadCopy nach einem Reset Ihres Rechners.

Tabelle 1. So lösen Sie das Speicherplatzproblem

1. Meldung ausgeben und Speicher reservieren. (»getmem«)
2. Mit Hilfe des »Trackdisk.device« werden die Tracks der Diskette geladen. (»losablock«)
3. Taskswitching abschalten. (»taskoff«)
4. Bitmap nach Block 879 verschieben
5. Blöcke ordnen. Schema:

RootBlock
File-Header
File-List
User-Dir
Datenblock
...
Datenblöcke
...
File-Header
File-List
Datenblock
...
Datenblöcke
...
und so weiter.

Die Hash-Tabelle des ersten User-Dirs beziehungsweise des Root-Blocks wird der Reihe nach durchsucht und deren File-Header oder User-Dirs hintenangestellt.

Hinter die File-Header kommen noch die File-Listblöcke und — falls nur ein Datenblock im File-Header eingetragen ist — dieser selbst. (Beispiel: ».info«-Datei für die Workbench). Darauf folgen die restlichen Datenblöcke des Directory. Dann sucht das Programm das nächste User-Dir und beginnt wieder von vorne. (»DiskSort« und »SeHashB«)

6. Diskette speichern (»Savedisk«)
7. "Maustastenrequester" ausgeben (»ende«)
8. Falls »Nochmal« und »Ramkopie« gewählt wurde: zurück nach »6«.
9. Bei »Nochmal« und »Diskkopie«: zurück nach »1«.
10. Bei »Quit« Speicher freigeben und »Ende« (»freemem«)

Tabelle 3. Eine Kurzübersicht des Programmes. Der Sourcecode von »FastLoadCopy« befindet sich auf der Programmservice-Diskette zu diesem Sonderheft.

Programmname:	FLC_GEN
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Generiert das lauffähige Maschinenprogramm »FLC« auf Diskette im Laufwerk df0:

```
Programmname: FLC_GEN
---------------------

  1 0m0 REM Generiert lauffähiges Programm
  2 ag  CLS
  3 ss  OPEN "df0:FLC" FOR OUTPUT AS 1
  4 BS  READ anz
  5 oa  FOR i=1 TO anz
  6 3n1  READ h$
  7 yB2   wert1=ASC(LEFT$(h$,1))
  8 bP    IF wert1>64 THEN wert1=wert1-87
           ELSE wert1=wert1-48
  9 FI    wert1=wert1*16
 10 7c    wert2=ASC(RIGHT$(h$,1))
 11 wp    IF wert2>64 THEN wert2=wert2-87
           ELSE wert2=wert2-48
 12 Pi    wert=wert1+wert2
 13 9G    PRINT #1,CHR$(wert);
 14 J00 NEXT
 15 3n  CLOSE 1
 16 0v  END
 17 yc  Werte:
 18 fR  DATA 5128
 19 ph  DATA 00,00,03,f3,00,00,00,00,00,00
 20 Qc  DATA 00,01,00,00,00,00,00,00,00,00
 21 A0  DATA 00,00,03,d5,00,00,03,e9,00,00
 22 vw  DATA 03,d5,23,cf,00,00,09,36,42,39
 23 4r  DATA 00,00,09,34,23,fc,00,00,0d,5c
 24 qN  DATA 00,00,0f,4a,23,fc,00,00,00,42
 25 1N  DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
 26 38  DATA 23,fc,00,00,0d,ba,00,00,0f,4a
 27 1G  DATA 23,fc,00,00,00,1c,00,00,0f,4e
 28 YJ  DATA 4e,b9,00,00,0c,f4,4e,b9,00,00
 29 sC  DATA 0b,34,23,fc,00,00,00,03,00,00
 30 Mw  DATA 0b,90,23,fc,00,06,e0,00,00,00
 31 Kz  DATA 0b,98,4e,b9,00,00,0b,5a,4a,b9
 32 KV  DATA 00,00,0b,94,67,00,09,04,23,f9
 33 sf  DATA 00,00,0b,94,00,00,0b,2c,23,fc
 34 2q  DATA 00,00,00,05,00,00,0b,90,4e,b9
 35 dz  DATA 00,00,0b,5a,4a,b9,00,00,0b,94
 36 p7  DATA 67,00,08,e0,23,f9,00,00,0b,94
 37 i5  DATA 00,00,0b,30,23,fc,00,00,0e,3c
 38 q7  DATA 00,00,0f,4a,23,fc,00,00,00,0d
 39 Fb  DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
 40 Lp  DATA 23,f9,00,00,0b,98,00,00,0c,e6
 41 gR  DATA 23,f9,00,00,0b,2c,00,00,0c,ea
 42 q2  DATA 33,fc,00,02,00,00,0c,f2,23,fc
 43 XM  DATA 00,00,03,70,00,00,0c,ee,4e,b9
 44 ge  DATA 00,00,0b,9c,bc,b9,00,00,0b,98
 45 8B  DATA 66,00,08,a6,4e,b9,00,00,0b,0e
 46 a2  DATA 23,fc,00,00,00,00,00,00,0c,ee
 47 71  DATA 4e,b9,00,00,0b,9c,bc,b9,00,00
 48 vh  DATA 0b,98,66,00,08,86,23,fc,00,00
 49 1c  DATA 0e,49,00,00,0f,4a,23,fc,00,00
 50 Rf  DATA 00,0e,00,00,0f,4e,4e,b9,00,00
 51 Bj  DATA 0c,f4,2c,79,00,00,00,04,4e,ae
 52 8T  DATA ff,7c,20,3c,00,00,1b,90,42,81
 53 gN  DATA 4e,ae,ff,3a,4a,80,67,00,08,30
 54 pz  DATA 23,c0,00,00,07,ba,20,79,00,00
 55 kS  DATA 0b,30,20,28,01,3c,33,c0,00,00
 56 kc  DATA 07,b6,21,7c,00,00,03,6f,01,3c
 57 Yt  DATA 4e,b9,00,00,05,04,23,c8,00,00
 58 Ij  DATA 05,00,22,3c,00,00,03,6f,74,01
 59 0K  DATA 4e,b9,00,00,01,fa,23,f9,00,00
 60 rN  DATA 07,ae,00,00,05,00,33,c1,00,00
 61 6S  DATA 07,b6,2e,3c,00,00,03,71,28,07
 62 QA  DATA 20,3c,00,00,03,70,42,86,7a,02
 63 Zr  DATA 33,c7,00,00,07,4a,4e,b9,00,00
 64 m6  DATA 05,36,20,04,52,84,b8,bc,00,00
 65 0J  DATA 06,e0,66,02,78,02,4e,b9,00,00
 66 h7  DATA 04,d6,66,20,b8,bc,00,00,03,6f
 67 Lq  DATA 67,18,4e,b9,00,00,05,04,24,10
 68 sZ  DATA b4,bc,00,00,00,08,67,d2,94,a8
 69 XV  DATA 01,fc,67,bc,60,ca,4e,b9,00,00
 70 v9  DATA 07,50,2c,79,00,00,00,04,22,79
 71 0j  DATA 00,00,07,ba,20,3c,00,00,1b,90
 72 Q4  DATA 4e,ae,ff,2e,60,00,08,94,48,e7
 73 gZ  DATA fe,fe,4e,b9,00,00,05,04,23,c8
 74 zx  DATA 00,00,07,ae,23,c0,00,00,07,a6
 75 Qo  DATA 23,c1,00,00,07,aa,20,01,4e,b9
 76 h5  DATA 00,00,05,04,23,c8,00,00,07,b2
 77 1J  DATA 4e,b9,00,00,02,a6,20,39,00,00
 78 9n  DATA 07,a6,23,f9,00,00,07,aa,00,00
 79 hS  DATA 07,a6,23,c0,00,00,07,aa,20,39
 80 MT  DATA 00,00,07,ae,23,f9,00,00,07,b2
 81 ql  DATA 00,00,07,ae,23,c0,00,00,07,b2
 82 LF  DATA 42,79,00,00,07,b8,20,7c,00,00
 83 H1  DATA 00,00,4e,b9,00,00,08,3a,20,39
 84 IT  DATA 00,00,07,a6,4e,b9,00,00,04,d6
 85 dS  DATA 66,18,20,79,00,00,07,ae,24,10
 86 PD  DATA b4,3c,00,08,67,04,94,a8,01,fc
 87 UP  DATA 4e,b9,00,00,02,a6,4e,b9,00,00
 88 E1  DATA 08,68,4e,b9,00,00,07,e2,4e,b9
 89 38  DATA 00,00,07,c0,4c,df,7f,7f,4e,75
 90 ns  DATA 2c,39,00,00,07,a6,b4,3c,00,05
 91 gG  DATA 67,2c,b4,3c,00,08,67,00,01,14
 92 Bq  DATA 4a,02,67,00,00,e8,b4,3c,00,13
 93 sx  DATA 67,00,00,9e,b4,3c,00,01,67,0e
 94 Zg  DATA 13,fc,00,08,00,00,09,34,4e,b9
 95 rp  DATA 00,00,08,b4,4e,75,48,e7,ff,fe
 96 iy  DATA 4e,b9,00,00,04,ac,42,85,20,28
 97 Hv  DATA 01,f4,4e,b9,00,00,05,36,22,48
 98 kV  DATA 20,28,00,10,4e,b9,00,00,05,04
 99 ZQ  DATA bc,a8,00,04,67,0e,13,fc,00,03
100 09  DATA 00,00,09,34,4e,b9,00,00,08,b4
101 xz  DATA 33,fc,00,04,00,00,07,b8,4e,b9
102 90  DATA 00,00,08,3a,20,28,00,10,4a,80
103 PP  DATA 66,d0,20,49,20,28,01,f8,4a,80
104 tK  DATA 67,2a,4e,b9,00,00,05,04,bc,a8
105 iU  DATA 01,f4,67,0e,13,fc,00,04,00,00
106 xt  DATA 09,34,4e,b9,00,00,08,b4,33,fc
107 X4  DATA 01,f4,00,00,07,b8,4e,b9,00,00
108 YF  DATA 08,3a,60,ce,4c,df,7f,ff,4e,75
109 Fl  DATA 48,e7,ff,fe,4e,b9,00,00,04,ac
110 Gg  DATA 20,28,01,f4,4e,b9,00,00,05,04
111 5m  DATA 20,28,01,f8,4a,80,66,0e,13,fc
112 1Y  DATA 00,05,00,00,09,34,4e,b9,00,00
113 Ix  DATA 08,b4,bc,80,66,e0,33,fc,01,f8
114 NU  DATA 00,00,07,b8,4e,b9,00,00,08,3a
115 V7  DATA 4c,df,7f,ff,4e,75,48,e7,ff,fe
116 Sg  DATA 4e,b9,00,00,04,ac,20,28,01,f4
117 LW  DATA 42,85,4e,b9,00,00,05,36,7a,01
118 3t  DATA 20,06,4e,b9,00,00,05,36,4c,df
119 jJ  DATA 7f,ff,4e,75,48,e7,ff,fe,20,79
120 i6  DATA 00,00,07,ae,20,28,00,04,22,28
121 f3  DATA 00,08,b2,7c,00,01,67,68,4e,b9
122 ms  DATA 00,00,05,04,d1,fc,00,00,00,18
123 gh  DATA 72,47,bc,98,57,c9,ff,fc,67,0e
124 2E  DATA 20,79,00,00,05,32,20,28,01,f8
125 To  DATA 24,48,60,dc,33,fc,ff,fc,00,00
126 Y3  DATA 07,b8,4e,b9,00,00,08,3a,4a,90
127 sD  DATA 67,04,20,10,60,04,20,2a,00,18
128 BB  DATA 4e,b9,00,00,05,04,bc,a8,00,10
129 xl  DATA 67,0e,13,fc,00,06,00,00,09,34
130 ec  DATA 4e,b9,00,00,08,b4,33,fc,00,10
131 el  DATA 00,00,07,b8,4e,b9,00,00,08,3a
132 Fc  DATA 60,5a,4e,b9,00,00,05,04,bc,a8
133 BA  DATA 01,34,67,0e,13,fc,00,06,00,00
134 PL  DATA 09,34,4e,b9,00,00,08,b4,33,fc
135 sa  DATA 01,34,00,00,07,b8,4e,b9,00,00
136 gk  DATA 08,3a,bc,a8,00,10,67,0e,13,fc
137 mp  DATA 00,01,00,00,09,34,4e,b9,00,00
138 No  DATA 08,b4,33,fc,00,10,00,00,07,b8
139 tT  DATA 4e,b9,00,00,08,3a,20,28,01,f8
140 T4  DATA 4a,80,67,08,4e,b9,00,00,05,04
141 cT  DATA 60,ce,4c,df,7f,ff,4e,75,20,79
142 q6  DATA 00,00,07,ae,bc,a8,00,04,67,0e
143 KQ  DATA 13,fc,00,07,00,00,09,34,4e,b9
144 in  DATA 00,00,08,b4,33,fc,00,04,00,00
145 Gs  DATA 07,b8,4e,b9,00,00,08,3a,4e,75
146 z4  DATA 48,e7,80,00,2a,39,00,00,05,00
147 Wd  DATA 72,02,58,85,06,41,00,20,b0,41
148 Eh  DATA 6a,f6,92,40,cb,88,2a,10,04,41
149 G4  DATA 00,20,44,41,03,05,4c,df,00,01
150 31  DATA 4e,75,00,00,00,00,48,e7,80,00
151 E1  DATA b0,7c,03,70,6a,08,20,79,00,00
152 RH  DATA 0b,2c,60,0a,20,79,00,00,0b,30
153 ul  DATA 04,40,03,70,c0,fc,02,00,d1,c0
154 Wp  DATA 23,c8,00,00,05,32,4c,df,00,01
155 85  DATA 4e,75,00,00,00,00,48,e7,f8,c0
156 uv  DATA 4e,b9,00,00,05,04,d1,fc,00,00
157 R0  DATA 00,18,76,47,20,18,4a,80,67,48
158 6p  DATA 4e,b9,00,00,05,d8,ba,bc,00,00
159 Eu  DATA 00,02,66,08,20,39,00,00,07,a6
160 2E  DATA 60,04,bc,80,67,44,22,48,4e,b9
161 pt  DATA 00,00,05,04,20,28,01,f0,4a,80
162 B3  DATA 67,1c,4e,b9,00,00,05,d8,ba,bc
163 qg  DATA 00,00,00,02,66,08,20,39,00,00
164 15  DATA 07,a6,60,dc,bc,80,67,20,60,d6
165 0H  DATA 20,49,51,cb,ff,b0,4a,85,66,26
166 jq  DATA 13,fc,00,08,00,00,09,34,4e,b9
167 tv  DATA 00,00,08,b4,91,fc,00,00,01,f4
168 1B  DATA 4a,85,66,e8,33,fc,01,f0,00,00
169 ST  DATA 07,b8,4e,b9,00,00,08,3a,ba,bc
170 Qb  DATA 00,00,00,02,66,06,4e,b9,00,00
171 M1  DATA 06,be,4c,df,03,1f,4e,75,4a,85
172 oE  DATA 67,3a,ba,bc,00,00,00,02,67,34
173 Uy  DATA 48,e7,80,80,4e,b9,00,00,05,04
174 CX  DATA 20,28,01,f4,bc,80,67,0e,13,fc
175 UT  DATA 00,02,00,00,09,34,4e,b9,00,00
176 yL  DATA 08,b4,33,fc,01,f4,00,00,07,b8
177 wp  DATA 4e,b9,00,00,08,3a,4c,df,01,01
178 oF  DATA 4e,75,48,e7,e0,80,4e,b9,00,00
179 jV  DATA 05,04,24,10,b4,3c,00,08,67,04
180 80  DATA 94,a8,01,fc,22,07,4e,b9,00,00
181 vY  DATA 01,fa,52,87,52,86,4e,b9,00,00
182 Mj  DATA 07,3e,b4,bc,00,00,00,05,66,70
183 Q3  DATA 23,f9,00,00,07,a6,00,00,07,4c
184 20  DATA 20,39,00,00,07,a6,4e,b9,00,00
185 eL  DATA 05,04,20,28,01,f8,4a,80,67,1a
186 TD  DATA 24,3c,00,00,00,13,22,07,4e,b9
187 03  DATA 00,00,01,fa,52,87,52,86,4e,b9
188 oI  DATA 00,00,07,3e,60,d2,b4,3c,00,13
189 Cm  DATA 67,28,20,28,00,08,b0,bc,00,00
190 2P  DATA 00,01,66,1c,20,28,00,10,22,07
191 MC  DATA 24,3c,00,00,00,08,4e,b9,00,00
192 6j  DATA 01,fa,52,87,52,86,4e,b9,00,00
193 ny  DATA 07,3e,23,f9,00,00,07,4c,00,00
194 09  DATA 07,a6,4c,df,01,07,4e,75,30,39
195 uW  DATA 00,00,07,4a,48,c0,4a,86,67,3a
196 0u  DATA 4e,b9,00,00,05,04,22,10,b2,bc
197 QX  DATA 00,00,00,08,67,1e,92,a8,01,fc
198 0u  DATA b2,bc,00,00,00,05,66,12,22,28
199 N0  DATA 00,08,b2,bc,00,00,00,01,67,06
200 vi  DATA 4e,b9,00,00,07,0a,52,80,b0,bc
201 uT  DATA 00,00,06,df,63,02,70,02,51,ce
202 Nn  DATA ff,c0,4e,75,48,e7,e0,00,20,28
203 2H  DATA 00,10,67,24,22,07,24,3c,00,00
204 JR  DATA 00,08,4e,b9,00,00,01,fa,52,87
205 1f  DATA 4e,b9,00,00,07,3e,20,39,00,00
206 60  DATA 07,a6,4e,b9,00,00,05,04,60,d6
207 SC  DATA 4c,df,00,07,4e,75,be,bc,00,00
208 YH  DATA 06,df,63,02,7e,02,4e,75,00,00
209 g7  DATA 00,00,00,00,70,01,52,40,b0,79
210 dd  DATA 00,00,07,b6,67,2c,b0,7c,06,e0
211 Mo  DATA 67,24,4e,b9,00,00,04,d6,66,e8
212 S1  DATA 4e,b9,00,00,05,04,42,84,76,7f
213 cW  DATA d8,98,51,cb,ff,fc,98,a8,fe,14
214 f7  DATA 44,84,21,44,fe,14,60,cc,4e,75
215 Ep  DATA 4e,b9,00,00,05,04,76,7e,d1,fc
216 Fm  DATA 00,00,00,04,42,84,d8,98,51,cb
217 NU  DATA ff,fc,44,84,21,44,fe,00,60,ac
218 Wm  DATA 00,00,00,00,00,00,00,00,00,00
219 Xn  DATA 00,00,00,00,00,00,00,00,00,00
220 tC  DATA 00,00,00,00,00,00,48,e7,c0,c0
221 3n  DATA 20,79,00,00,07,ae,22,79,00,00
222 bt  DATA 07,b2,70,7f,22,10,20,d1,22,c1
223 Nf  DATA 51,c8,ff,f8,4c,df,03,03,4e,75
224 al  DATA 48,e7,fe,fe,20,39,00,00,07,a6
225 rD  DATA 4e,b9,00,00,04,d6,2c,48,28,01
226 b3  DATA 2c,05,20,39,00,00,07,aa,4e,b9
227 Gt  DATA 00,00,04,d6,bc,85,67,1c,09,06
```

Listing 1. Der Basic-Lader von »Fast LoadCopy«

```
228 Pa  DATA 67,08,03,c5,67,08,09,c6,60,06
229 wB  DATA 03,85,60,f6,09,86,2c,86,20,85
230 g4  DATA 4c,df,7f,7f,4e,75,09,06,67,08
231 Co  DATA 03,c6,67,08,09,c6,60,06,03,86
232 9e  DATA 60,f6,09,86,2c,86,60,e4,48,e7
233 Pn  DATA 80,c0,30,39,00,00,07,b8,48,c0
234 nd  DATA d1,c0,30,39,00,00,07,be,48,c0
235 JV  DATA 22,79,00,00,07,ba,d3,c0,22,88
236 XQ  DATA 58,80,33,c0,00,00,07,be,4c,df
237 SM  DATA 03,01,4e,75,36,39,00,00,07,be
238 1x  DATA e4,4b,53,43,20,79,00,00,07,ba
239 0b  DATA 22,39,00,00,07,a6,4a,90,67,20
240 vU  DATA 22,58,22,81,51,cb,ff,f6,42,79
241 55  DATA 00,00,07,be,20,79,00,00,07,ba
242 wv  DATA 36,3c,06,e3,42,98,51,cb,ff,fc
243 Ys  DATA 4e,75,d1,fc,00,00,00,04,22,39
244 4P  DATA 00,00,07,aa,51,cb,ff,ce,60,d6
245 zB  DATA 48,e7,ff,fe,20,39,00,00,07,a6
246 eK  DATA 41,f9,00,00,0f,1e,76,02,4e,b9
247 bl  DATA 00,00,09,3a,41,f9,00,00,0f,48
248 rU  DATA 10,39,00,00,09,34,48,80,42,83
249 pD  DATA 4e,b9,00,00,09,3a,2c,79,00,00
250 CI  DATA 00,04,4e,ae,ff,76,23,fc,00,00
251 uw  DATA 0f,0d,00,00,0f,4a,23,fc,00,00
252 qB  DATA 00,3d,00,00,0f,4e,4e,b9,00,00
253 LB  DATA 0c,f4,10,39,00,00,09,34,b0,3c
254 gJ  DATA 00,07,63,16,4e,b9,00,00,07,50
255 WS  DATA 4e,b9,00,00,0b,0e,2e,79,00,00
256 2v  DATA 09,36,60,00,f8,ba,42,39,00,00
257 ku  DATA 09,34,4c,df,7f,ff,4e,75,00,00
258 SP  DATA 00,00,00,00,48,e7,f0,80,d1,c3
259 V7  DATA d1,fc,00,00,01,42,42,34,00
260 w3  DATA 02,42,00,0f,06,42,00,30,b4,7c
261 ZM  DATA 00,39,63,00,00,06,06,42,00,07
262 10  DATA 11,02,e8,48,51,cb,ff,e2,4c,df
263 52  DATA 01,0f,4e,75,23,fc,00,00,0d,9e
264 Tk  DATA 00,00,0f,4a,23,fc,00,00,00,1c
265 tF  DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
266 1f  DATA 4e,f9,00,00,0a,4a,23,fc,00,00
267 Yg  DATA 0e,14,00,00,0f,4a,23,fc,00,00
268 qD  DATA 00,1c,00,00,0f,4e,4e,b9,00,00
269 fF  DATA 0c,f4,4e,f9,00,00,0a,4a,23,fc
270 xq  DATA 00,00,0e,57,00,00,0f,4a,23,fc
271 08  DATA 00,00,00,5c,00,00,0f,4e,4e,b9
272 uG  DATA 00,00,0c,f4,42,00,4e,b9,00,00
273 vd  DATA 0a,2c,4a,00,67,00,00,78,23,fc
274 FA  DATA 00,00,0e,b3,00,00,0f,4a,23,fc
275 06  DATA 00,00,00,5a,00,00,0f,4e,4e,b9
276 9d  DATA 00,00,0c,f4,08,39,00,0a,00,df
277 3D  DATA f0,16,67,00,ff,f6,42,00,4e,b9
278 GP  DATA 00,00,0a,2c,4a,00,67,00,00,7a
279 mA  DATA 23,fc,00,00,0d,5c,00,00,0f,4a
280 WK  DATA 23,fc,00,00,00,42,00,00,0f,4e
281 d0  DATA 4e,b9,00,00,0c,f4,4e,b9,00,00
282 tc  DATA 0b,0e,60,00,00,76,08,39,00,06
283 j1  DATA 00,bf,e0,01,67,00,00,12,08,39
284 Je  DATA 00,0a,00,df,f0,16,66,00,ff,ea
285 zK  DATA 06,00,00,01,4e,75,4a,b9,00,00
286 n1  DATA 0b,30,67,00,00,12,23,f9,00,00
287 Gz  DATA 0b,30,00,00,0b,94,4e,b9,00,00
288 vi  DATA 0b,78,4a,b9,00,00,0b,2c,67,00
289 yU  DATA 00,12,23,f9,00,00,0b,2c,00,00
290 bC  DATA 0b,94,4e,b9,00,00,0b,78,4e,75
291 06  DATA 4e,b9,00,00,0a,4a,4e,f9,00,00
292 ry  DATA 00,00,2c,79,00,00,00,04,4e,ae
293 Dz  DATA ff,76,4a,39,00,00,09,34,66,00
294 Ku  DATA ff,0e,23,fc,00,00,0d,d6,00,00
295 wE  DATA 0f,4a,23,fc,00,00,00,1c,00,00
296 RF  DATA 0f,4e,4e,b9,00,00,0c,f4,4e,b9
297 vX  DATA 00,00,0b,34,23,fc,00,00,0e,30
298 xD  DATA 00,00,0f,4a,23,fc,00,00,00,0c
299 Rn  DATA 00,00,0f,4e,4e,b9,00,00,0c,f4
300 YW  DATA 33,fc,00,0b,00,00,0c,f2,23,fc
301 p3  DATA 00,00,00,00,00,00,0c,ee,4e,b9
302 5v  DATA 00,00,0b,9c,4e,b9,00,00,0b,0e
303 Co  DATA 23,fc,00,00,03,70,00,00,0c,ee
304 wm  DATA 4e,b9,00,00,0b,9c,4e,f9,00,00
305 2S  DATA 09,ac,22,79,00,00,0b,2c,20,79
306 J0  DATA 00,00,0b,30,20,3c,00,01,b8,00
307 An  DATA 22,10,20,d1,22,c1,53,80,66,f6
308 ig  DATA 4e,75,00,00,00,00,00,00,00,00
309 Tr  DATA 23,fc,00,00,0d,f2,00,00,0f,4a
310 wi  DATA 23,fc,00,00,00,22,00,00,0f,4e
311 ea  DATA 4e,b9,00,00,0c,f4,08,39,00,06
312 vK  DATA 00,bf,e0,01,66,f6,4e,75,2c,79
313 gJ  DATA 00,00,00,04,20,39,00,00,0b,98
314 1H  DATA 22,39,00,00,0b,90,4e,ae,ff,3a
315 31  DATA 23,c0,00,00,0b,94,4e,75,2c,79
316 JM  DATA 00,00,00,04,20,39,00,00,0b,98
317 qH  DATA 22,79,00,00,0b,94,4e,ae,ff,2e
318 sq  DATA 4e,75,00,00,00,00,00,00,00,00
319 MZ  DATA 00,00,00,00,4d,f9,00,00,0c,76
320 rA  DATA 20,3c,00,00,00,1b,2c,fc,00,00
321 fb  DATA 00,00,51,c8,ff,f8,2c,79,00,00
322 1W  DATA 00,04,20,39,00,00,0c,ee,c0,fc
323 RN  DATA 02,00,23,c0,00,00,0c,ee,93,c9
324 kK  DATA 4e,ae,fe,da,43,f9,00,00,0c,c6
325 ak  DATA 23,40,00,10,4e,ae,fe,9e,43,f9
326 7k  DATA 00,00,0c,76,20,3c,00,00,00,00
327 2J  DATA 42,81,41,f9,00,00,0c,64,4e,ae
328 ST  DATA fe,44,4a,80,66,00,00,6a,43,f9
329 xR  DATA 00,00,0c,76,23,7c,00,00,0c,c6
330 ch  DATA 00,0e,33,79,00,00,0c,f2,00,1c
331 Gw  DATA 23,79,00,00,0c,ea,00,28,23,79
332 bY  DATA 00,00,0c,e6,00,24,23,79,00,00
333 Jh  DATA 0c,ee,00,2c,2c,79,00,00,00,04
334 ON  DATA 4e,ae,fe,38,43,f9,00,00,0c,76
335 3V  DATA 2c,29,00,20,33,7c,00,09,00,1c
336 1W  DATA 23,7c,00,00,00,00,00,24,4e,ae
337 z9  DATA fe,38,43,f9,00,00,0c,c6,4e,ae
338 Mu  DATA fe,98,43,f9,00,00,0c,76,4e,ae
339 fu  DATA fe,3e,4e,75,74,72,61,63,6b,64
340 1D  DATA 69,73,6b,2e,64,65,76,69,63,65
341 V1  DATA 00,00,00,00,00,00,00,00,00,00
342 Wm  DATA 00,00,00,00,00,00,00,00,00,00
343 Xn  DATA 00,00,00,00,00,00,00,00,00,00
344 Yo  DATA 00,00,00,00,00,00,00,00,00,00
345 Zp  DATA 00,00,00,00,00,00,00,00,00,00
346 aq  DATA 00,00,00,00,00,00,00,00,00,00
347 br  DATA 00,00,00,00,00,00,00,00,00,00
348 cs  DATA 00,00,00,00,00,00,00,00,00,00
349 dt  DATA 00,00,00,00,00,00,00,00,00,00
350 eu  DATA 00,00,00,00,00,00,00,00,00,00
351 fv  DATA 00,00,00,00,00,00,00,00,00,00
352 gw  DATA 00,00,00,00,00,00,00,00,00,00
353 KQ  DATA 00,00,00,00,00,00,00,00,48,e7
354 of  DATA fe,fe,2c,79,00,00,00,04,43,f9
355 0m  DATA 00,00,0d,4c,42,80,4e,ae,fe,68
356 op  DATA 67,00,00,3a,23,c0,00,00,0d,58
357 xP  DATA 2c,40,4e,ae,ff,c4,22,00,24,39
358 eQ  DATA 00,00,0f,4a,26,39,00,00,0f,4e
359 sp  DATA 4e,ae,ff,d0,4a,80,6a,00,00,06
360 uz  DATA 4e,ee,ff,70,2c,79,00,00,00,04
361 ta  DATA 22,79,00,00,0d,58,4e,ae,fe,62
362 vE  DATA 4c,df,7f,7f,4e,75,64,6f,73,2e
363 GS  DATA 6c,69,62,72,61,72,79,00,00,00
364 D4  DATA 00,00,0c,0a,0a,09,09,9b,31,3b
365 ux  DATA 33,31,3b,34,32,6d,20,20,20,20
366 mo  DATA 46,61,73,74,4c,6f,61,64,43,6f
367 Tv  DATA 70,79,20,62,79,20,50,2e,53,2e
368 S9  DATA 20,20,20,28,43,29,20,4d,26,54
369 5a  DATA 20,69,6e,20,31,39,38,38,20,20
370 v1  DATA 20,20,9b,30,6d,0a,0a,0a,5a,75
371 uq  DATA 20,77,65,6e,69,67,20,53,70,65
372 KK  DATA 69,63,68,65,72,20,76,6f,72,68
373 Am  DATA 61,6e,64,65,6e,0a,42,69,74,74
374 1w  DATA 65,20,51,75,65,6c,6c,64,69,73
375 J8  DATA 6b,65,74,74,65,20,69,6e,20,44
376 Lo  DATA 46,30,3a,0a,0a,42,69,74,74,65
377 fA  DATA 20,5a,69,65,6c,64,69,73,6b,65
378 X7  DATA 74,74,65,20,69,6e,20,44,46,30
379 oJ  DATA 3a,0a,2e,2e,2e,20,75,6e,64,20
380 92  DATA 6c,69,6e,6b,65,20,4d,61,75,73
381 JY  DATA 74,61,73,74,65,20,64,72,75,65
382 oJ  DATA 63,6b,65,6e,21,0a,54,72,61,63
383 pM  DATA 6b,64,69,73,6b,2e,64,65,76,69
384 Rz  DATA 63,65,20,2d,20,50,72,6f,62,6c
385 1I  DATA 65,6d,65,0a,2e,2e,2e,20,73,61
386 mH  DATA 76,69,6e,67,0a,0a,2e,2e,2e,20
387 rk  DATA 6c,6f,61,64,69,6e,67,0a,0a,2e
388 7V  DATA 2e,2e,20,63,68,61,6e,67,69,6e
389 fA  DATA 67,0a,0a,0a,0a,0a,09,9b,30,3b
390 T5  DATA 33,31,3b,34,32,6d,20,20,4c,4d
391 vv  DATA 61,75,73,74,61,73,74,65,20,3d
392 1I  DATA 20,20,20,51,75,69,74,20,20,20
393 9t  DATA 20,9b,30,6d,20,20,20,20,20,20
394 PL  DATA 20,20,20,20,20,20,20,9b,30,3b
395 4p  DATA 33,31,3b,34,32,6d,20,20,52,4d
396 00  DATA 61,75,73,74,61,73,74,65,20,3d
397 0Q  DATA 20,20,4e,6f,63,68,6d,61,6c,20
398 p9  DATA 20,9b,30,6d,0a,0a,09,9b,30,3b
399 cE  DATA 33,31,3b,34,32,6d,20,20,4c,4d
400 44  DATA 61,75,73,74,61,73,74,65,20,3d
401 d2  DATA 20,44,69,73,6b,6b,6f,70,69,65
402 I2  DATA 20,9b,30,6d,20,20,20,20,20,20
403 YU  DATA 20,20,20,20,20,20,20,9b,30,3b
404 Dy  DATA 33,31,3b,34,32,6d,20,20,52,4d
405 99  DATA 61,75,73,74,61,73,74,65,20,3d
406 M7  DATA 20,52,61,6d,6b,6f,70,69,65,20
407 12  DATA 20,9b,30,6d,0a,46,65,68,6c,65
408 sU  DATA 72,20,69,6e,20,42,6c,6f,63,6b
409 RP  DATA 20,24,20,20,20,2e,20,44,69,73
410 vE  DATA 6b,20,75,2e,55,2e,20,66,65,68
411 0y  DATA 6c,65,72,68,61,66,74,20,21,21
412 tN  DATA 20,46,65,68,6c,65,72,6e,75,6d
413 VJ  DATA 6d,65,72,20,20,0a,00,00,00,00
414 JI  DATA 00,00,00,00,00,00,00,00,03,ec
415 x6  DATA 00,00,01,20,00,00,00,00,00,00
416 Zg  DATA 00,02,00,00,00,08,00,00,00,0e
417 EI  DATA 00,00,00,12,00,00,00,1c,00,00
418 ny  DATA 00,22,00,00,00,28,00,00,00,2c
419 gi  DATA 00,00,00,36,00,00,00,3c,00,00
420 hx  DATA 00,42,00,00,00,4c,00,00,00,56
421 LE  DATA 00,00,00,5c,00,00,00,62,00,00
422 rw  DATA 00,6c,00,00,00,70,00,00,00,7a
423 NL  DATA 00,00,00,80,00,00,00,86,00,00
424 a8  DATA 00,90,00,00,00,94,00,00,00,9a
425 Km  DATA 00,00,00,9e,00,00,00,a8,00,00
426 d4  DATA 00,ae,00,00,00,b4,00,00,00,b8
427 2m  DATA 00,00,00,be,00,00,00,c2,00,00
428 w0  DATA 00,ca,00,00,00,d4,00,00,00,da
429 mG  DATA 00,00,00,e0,00,00,00,ea,00,00
430 pN  DATA 00,f4,00,00,00,fa,00,00,01,00
431 7y  DATA 00,00,01,0a,00,00,01,0e,00,00
432 qa  DATA 01,18,00,00,01,1e,00,00,01,40
433 y6  DATA 00,00,01,46,00,00,01,50,00,00
434 hy  DATA 01,5e,00,00,01,64,00,00,01,72
435 kZ  DATA 00,00,01,78,00,00,01,7c,00,00
436 rt  DATA 01,82,00,00,01,9a,00,00,01,a0
437 uq  DATA 00,00,01,b4,00,00,01,c4,00,00
438 UL  DATA 01,dc,00,00,01,e8,00,00,02,00
439 ty  DATA 00,00,02,06,00,00,02,0c,00,00
440 EJ  DATA 02,12,00,00,02,1a,00,00,02,20
441 DC  DATA 00,00,02,26,00,00,02,2c,00,00
442 as  DATA 02,32,00,00,02,36,00,00,02,3c
443 d1  DATA 00,00,02,42,00,00,02,48,00,00
444 yQ  DATA 02,4c,00,00,02,52,00,00,02,58
445 f0  DATA 00,00,02,64,00,00,02,6a,00,00
446 An  DATA 02,70,00,00,02,78,00,00,02,8a
447 E3  DATA 00,00,02,90,00,00,02,96,00,00
448 hh  DATA 02,9c,00,00,02,a8,00,00,02,d2
449 gX  DATA 00,00,02,d8,00,00,02,e4,00,00
450 8r  DATA 02,f0,00,00,02,fc,00,00,03,0a
451 Qc  DATA 00,00,03,10,00,00,03,18,00,00
452 xE  DATA 03,1e,00,00,03,36,00,00,03,44
453 8t  DATA 00,00,03,4a,00,00,03,52,00,00
454 KM  DATA 03,58,00,00,03,6a,00,00,03,74
455 Hr  DATA 00,00,03,84,00,00,03,8a,00,00
456 Bv  DATA 03,96,00,00,03,9c,00,00,03,ac
457 YQ  DATA 00,00,03,b8,00,00,03,c2,00,00
458 rc  DATA 03,d2,00,00,03,e6,00,00,03,fc
459 so  DATA 00,00,04,0c,00,00,04,12,00,00
460 e8  DATA 04,24,00,00,04,32,00,00,04,38
461 31  DATA 00,00,04,40,00,00,04,46,00,00
462 jr  DATA 04,4e,00,00,04,5c,00,00,04,62
463 cC  DATA 00,00,04,6a,00,00,04,70,00,00
464 uC  DATA 04,7e,00,00,04,84,00,00,04,8c
465 S1  DATA 00,00,04,92,00,00,04,a0,00,00
466 m5  DATA 04,ae,00,00,04,bc,00,00,04,c2
467 Dc  DATA 00,00,04,ca,00,00,04,d0,00,00
468 BN  DATA 04,dc,00,00,05,10,00,00,05,18
469 G9  DATA 00,00,05,28,00,00,05,3c,00,00
470 1S  DATA 05,50,00,00,05,5e,00,00,05,6c
471 ZB  DATA 00,00,05,7a,00,00,05,88,00,00
472 10  DATA 05,a2,00,00,05,a8,00,00,05,ba
473 Gf  DATA 00,00,05,c0,00,00,05,ce,00,00
474 cU  DATA 05,ea,00,00,05,fa,00,00,06,00
475 JG  DATA 00,00,06,08,00,00,06,0e,00,00
476 MS  DATA 06,1e,00,00,06,32,00,00,06,3c
477 Ff  DATA 00,00,06,4a,00,00,06,4e,00,00
478 So  DATA 06,54,00,00,06,5a,00,00,06,70
479 KQ  DATA 00,00,06,7a,00,00,06,a0,00,00
```

```
480 ZQ  DATA 06,aa,00,00,06,b0,00,00,06,b4
481 Qk  DATA 00,00,06,c0,00,00,06,cc,00,00
482 o0  DATA 06,f4,00,00,07,1e,00,00,07,26
483 2j  DATA 00,00,07,2c,00,00,07,32,00,00
484 oB  DATA 07,56,00,00,07,64,00,00,07,6c
485 AH  DATA 00,00,07,8a,00,00,07,c6,00,00
486 od  DATA 07,cc,00,00,07,e8,00,00,07,ee
487 t3  DATA 00,00,07,fa,00,00,08,00,00,00
488 T9  DATA 08,40,00,00,08,4a,00,00,08,52
489 p7  DATA 00,00,08,5e,00,00,08,6a,00,00
490 NJ  DATA 08,74,00,00,08,7a,00,00,08,8c
491 ga  DATA 00,00,08,92,00,00,08,aa,00,00
492 ae  DATA 08,ba,00,00,08,c0,00,00,08,c8
493 f0  DATA 00,00,08,ce,00,00,08,d4,00,00
494 VB  DATA 08,de,00,00,08,ee,00,00,08,f2
495 NY  DATA 00,00,08,fc,00,00,09,02,00,00
496 TX  DATA 09,08,00,00,09,14,00,00,09,1a
497 vR  DATA 00,00,09,20,00,00,09,2a,00,00
498 zR  DATA 09,6e,00,00,09,72,00,00,09,7c
499 Fq  DATA 00,00,09,82,00,00,09,88,00,00
500 Nv  DATA 09,8e,00,00,09,92,00,00,09,9c
501 Eq  DATA 00,00,09,a2,00,00,09,a8,00,00
502 gD  DATA 09,ae,00,00,09,b2,00,00,09,bc
503 CN  DATA 00,00,09,c2,00,00,09,ca,00,00
504 9q  DATA 09,d6,00,00,09,da,00,00,09,e4
505 LG  DATA 00,00,09,ea,00,00,09,fe,00,00
506 5Y  DATA 0a,0a,00,00,0a,0e,00,00,0a,18
507 F1  DATA 00,00,0a,1e,00,00,0a,24,00,00
508 5X  DATA 0a,4c,00,00,0a,56,00,00,0a,5a
509 N4  DATA 00,00,0a,60,00,00,0a,66,00,00
510 WL  DATA 0a,70,00,00,0a,74,00,00,0a,7a
511 tY  DATA 00,00,0a,82,00,00,0a,88,00,00
512 zx  DATA 0a,98,00,00,0a,a2,00,00,0a,a6
513 p0  DATA 00,00,0a,b0,00,00,0a,b6,00,00
514 gb  DATA 0a,bc,00,00,0a,c2,00,00,0a,c6
515 9c  DATA 00,00,0a,d0,00,00,0a,d6,00,00
516 L5  DATA 0a,de,00,00,0a,e8,00,00,0a,ee
517 dp  DATA 00,00,0a,f4,00,00,0a,fe,00,00
518 FT  DATA 0b,04,00,00,0b,0a,00,00,0b,10
519 GH  DATA 00,00,0b,16,00,00,0b,36,00,00
520 yW  DATA 0b,3a,00,00,0b,44,00,00,0b,4a
521 vd  DATA 00,00,0b,62,00,00,0b,68,00,00
522 VW  DATA 0b,72,00,00,0b,80,00,00,0b,86
523 fN  DATA 00,00,0b,9e,00,00,0b,ba,00,00
524 K6  DATA 0b,c4,00,00,0b,d0,00,00,0b,de
525 D9  DATA 00,00,0b,ec,00,00,0b,fc,00,00
526 Vw  DATA 0c,02,00,00,0c,0a,00,00,0c,12
527 dD  DATA 00,00,0c,1a,00,00,0c,22,00,00
528 oH  DATA 0c,34,00,00,0c,50,00,00,0c,5a
529 yj  DATA 00,00,0d,00,00,00,0d,10,00,00
530 sF  DATA 0d,1e,00,00,0d,24,00,00,0d,3e
531 fV  DATA 00,00,00,00,00,00,03,f2
(C) 1988 M&T
```

Listing 1. (Schluß)

Programmname:	FastLoadCopy
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Assembler
Assembler:	DevPac

```
Programmname: FastLoadCopy
--------------------------

  1 XS0  ;********************************
         ********************************
  2 Kp   ;*                FastLoadCopy b
         y Paul Sponagl                *
  3 s3   ;*                  (C) Markt &
          Technik                      *
  4 No   ;*                      in July
          1988                         *
  5 59   ;*                      TEL: 083
         68/1581                       *
  6 cX   ;********************************
         ********************************
  7 DS   ;****************** Alle Makros d
         effinieren ********************
  8 i1   ExecBase   equ 4
  9 jo   FindTask   equ -294
 10 Fy   AddPort    equ -354
 11 Vg   RemPort    equ -360
 12 fy   OpenLib    equ -408
              ;Exec-Kommandos
 13 b6   CloseLib   equ -414
 14 Y3   Forbit     equ -132
 15 Iy   Permit     equ -138
 16 nU   AllocMem   equ -198
 17 bm   FreeMem    equ -210
 18 tD   ChipMem    equ 3
              ; " (Speicher)
 19 X8   FastMem    equ 5
 20 Qf   OpenDev    equ -444
 21 ST   CloseDev   equ -450
              ; " (Trackdisk)
 22 Wy   DoIo       equ -456
 23 Xo   td_read    equ 2
 24 xL   td_format  equ 11
              ;Trackdisk-Kommandos
 25 Ae   OutPut     equ -60
 26 HH   Write      equ -48
              ;DOS-Kommandos
 27 KT   Exit       equ -144
 28 1k   ;************************** Start
          ******************************
 29 9n   start:
 30 Tw5      move.l a7,SPointer
 31 jP       clr.b  Errorstate
 32 mv       move.l #clstxt,text
 33 4r       move.l #memtxt-clstxt,textl
             en     ;Anfangstext
 34 BK       jsr    printtxt
 35 dQ       move.l #quelltxt,text
 36 s8       move.l #zieltxt-quelltxt,te
             xtlen  ;Quelldiskette!
 37 EN       jsr    printtxt
 38 Ep       jsr    wait
 39 M90  ;************ Ganze Diskette in
         den Speicher laden ************
 40 lg5      move.l #ChipMem,Bedingung
 41 gr       move.l #450560,MemLaenge
                    ;ChipMem fuer 880 Bloe
                    cke
 42 wd       jsr    getmem
                   ;reservieren
 43 kl       tst.l  Buffer
 44 RU       beq    memfehler
 45 gV       move.l Buffer,Chipbuffer
 46 wa       move.l #FastMem,Bedingung
 47 SX       jsr    getmem
                   ;FastMem ebenfalls
 48 pq       tst.l  Buffer
 49 WZ       beq    memfehler
 50 ZJ       move.l Buffer,Fastbuffer
 51 Tx       move.l #ltxt,text
 52 nH       move.l #ctxt-ltxt,textlen
                    ;"loading"ausgeben
 53 Ud       jsr    printtxt
 54 HJ       move.l MemLaenge,bufflen
 55 do       move.l Chipbuffer,diskbuff
 56 zr       move   #td_read,Kommando
                    ;Block 880-1759 laden
 57 Xu       move.l #880,Blk_Nr
 58 zH       jsr    losablock
 59 mX       cmp.l  MemLaenge,d6
 60 HQ       bne    trackfehler
 61 wB       jsr    swapbuffer
                    ;Fastmem<->Chipmem
 62 1v       move.l #0,Blk_Nr
 63 Az       jsr    losablock
                    ;Block 0-879 laden
 64 rc       cmp.l  MemLaenge,d6
 65 MV       bne    trackfehler
 66 ez0  ;**************** Blockchangerou
         tine *************************
 67 OL5      move.l #ctxt,text
 68 L1       move.l #endtxt1-ctxt,textle
             n      ;Changing ausgeben
 69 kt       jsr    printtxt
 70 gn0  taskoff:
 71 zC5      move.l ExecBase,a6
                    ;Taskswitching
 72 vQ       jsr    Forbit(a6)
                    ;verbieten
 73 01       move.l #7056,d0
 74 CF       clr.l  d1
 75 3W       jsr    AllocMem(a6)
 76 Fi       tst.l  d0
 77 y1       beq    memfehler
 78 tI       move.l d0,Listadr
 79 Fy       move.l Fastbuffer,a0
 80 1A       move.l 316(a0),d0
 81 6B       move   d0,BitmapBlock
                    ;Adresse des Bitmapbloc
                    kes
 82 Sw       move.l #879,316(a0)
 83 nS       jsr    AdrRech
                    ;feststellen
 84 Xn       move.l a0,BMADR
 85 J1       move.l #879,d1
 86 Z0       moveq  #1,d2
 87 WX       jsr    BlockMove
                    ;Bitmap nach Block 881
 88 yb       move.l ABADR,BMADR
 89 J4       move   d1,BitmapBlock
 90 Q00  DiskSort:
 91 fS5      move.l #881,d7
 92 JR       move.l d7,d4
 93 5o       move.l #880,d0
                    ;RootBlock sortieren
 94 3S0  DSSLoop:
 95 Xa5      clr.l  d6
 96 oG       moveq  #2,d5
 97 3V       move   d7,Firstsub
 98 1W       jsr    SeHashB
 99 g00  keinUDDS:
100 CH5      move.l d4,d0
101 fB       addq.l #1,d4
102 10       cmp.l  #1760,d4
103 gh       bne.s  weiterUDDS
104 w0       moveq  #2,d4
105 N10  weiterUDDS:
106 aE5      jsr    SBitMap
107 I8       bne.s  FertigDiskS
108 Pz       cmp.l  #879,d4
109 fY       beq.s  FertigDiskS
110 IU       jsr    AdrRech
111 aV       move.l (a0),d2
112 pW       cmp.l  #8,d2
113 Sh       beq.s  keinUDDS
114 ru       sub.l  508(a0),d2
115 Zg       beq.s  DSSLoop
                    ;UserDir sortieren
116 6I       bra.s  keinUDDS
117 vq0  FertigDiskS:
118 ep5      jsr    ChkSumm
119 HB0  exitchange:
120 Ew5      move.l ExecBase,a6
121 31       move.l Listadr,a1
122 nV       move.l #7056,d0
123 Qq       jsr    FreeMem(a6)
124 YI       bra    taskan
125 cD0  ;******* BlockMove: in d0 alte Bl
         ocknummer ******
```

Listing 2.
Der Quelltext von FastLoadCopy

```
126 rI  ;                    in d1 neue Bl
        ocknummer      *
127 Jm  ;                    in d2 Blockar
        t              *
128 6r  ;********************************
        ****************
129 On  BlockMove:
130 t15      movem.l d0-d6/a0-a6,-(a7)
131 dp       jsr    AdrRech
132 ko       move.l a0,ABADR
133 Or       move.l d0,ABNR
                    ;Adressen fuer Ver-
134 rf       move.l d1,NBNR
                    ;schiebungsroutinen
135 WY       move.l d1,d0
136 iu       jsr    AdrRech
137 FW       move.l a0,NBADR
138 JE       jsr    WhatMove
                    ;1.Block(d0)
139 Nv       move.l ABNR,d0
140 Aw       move.l NBNR,ABNR
141 RO       move.l d0,NBNR
142 c0       move.l ABADR,d0
                    ;Adressen und
143 ee       move.l NBADR,ABADR
                    ;Nummern austauschen
144 Ys       move.l d0,NBADR
145 Oy       clr    offset
146 KF       move.l #0,a0
147 n2       jsr    Listeintrag
148 W4       move.l ABNR,d0
149 Hv       jsr    SBitMap
150 8U       bne.s  blockleer
                    ;Tauschblock leer?
151 KF       move.l ABADR,a0
152 FA       move.l (a0),d2
153 Wr       cmp.b  #8,d2
154 fA       beq.s  weiterBM
155 WZ       sub.l  508(a0),d2
156 1D0 weiterBM:
157 kc5      jsr    WhatMove
                    ;2.Block(d1)
158 yY0 blockleer:
159 ma5      jsr    Blockeintrag
160 tY       jsr    SwapBit
                    ;Blocks u. BBits austau
                    schen
161 sm       jsr    SwapBlock
162 h9       movem.l (a7)+,d0-d6/a0-a6
163 L1       rts
164 O10 WhatMove:
165 nL5      move.l ABNR,d6
166 Um       cmp.b  #5,d2
167 If       beq.s  FHMove
168 16       cmp.b  #8,d2
169 Yv       beq    DBMove
170 u0       tst.b  d2
                    ;Blockroutine auswaehle
                    n
171 qW       beq    UDMove
172 0a       cmp.b  #19,d2
173 Y7       beq    FLMove
174 IW       cmp.b  #1,d2
175 hX       beq.s  weiterWM
176 Vf       move.b #8,Errorstate
177 LB       jsr    changeerror
178 Ov0 weiterWM:
179 bH5      rts
180 t90 ;********** Blockerschiebungsrout
        inen ************
181 pa  ;* benoetigen die Adressen NBNR,A
        BNR,ABADR        *
182 mh  ;********************************
        ****************
183 mQ  ;************* File-Header versch
        . ***************
184 xm  FHMove:
185 qz5      movem.l d0-d7/a0-a6,-(a7)
186 6t       jsr    HKey
                    ;Neuer Header-Key
187 14       clr.l  d5
188 ym       move.l 500(a0),d0
189 Ez       jsr    SeHashB
190 2x       move.l a0,a1
191 xf       move.l 16(a0),d0
192 WE0 FHLoop1:
193 dp5      jsr    AdrRech
194 6z       cmp.l  4(a0),d6
195 NF       beq.s  noerrorFH2
196 QV       move.b #3,Errorstate
197 06       jsr    changeerror
                    ;In alle Datenbl.
198 Tq0 noerrorFH2:
199 QS5      move   #4,offset
200 et       jsr    Listeintrag
201 7p       move.l 16(a0),d0
202 Hk       tst.l  d0
203 Gp       bne.s  FHLoop1
204 LH       move.l a1,a0
205 mV0 FHLoop2:
206 eC5      move.l 504(a0),d0
207 Mp       tst.l  d0
208 7i       beq.s  fertigFH
209 t5       jsr    AdrRech
210 d2       cmp.l  500(a0),d6
211 Pi       beq.s  noerrorFH
                    ;In alle FileList
212 1r       move.b #4,Errorstate
213 v1       jsr    changeerror
214 zW0 noerrorFH:
215 x75      move   #500,offset
216 u9       jsr    Listeintrag
217 Qz       bra.s  FHLoop2
218 xZ0 fertigFH:
219 hA5      movem.l (a7)+,d0-d7/a0-a6
220 Gw       rts
221 iV0 ;************** File-List versch.
         ****************
222 le  FLMove:
223 Sb5      movem.l d0-d7/a0-a6,-(a7)
224 iV       jsr    HKey
                    ;Neuer Header-Key
225 ZN       move.l 500(a0),d0
226 Rw0 FLLoop:
227 BN5      jsr    AdrRech
228 OY       move.l 504(a0),d0
229 iB       tst.l  d0
230 JU       bne.s  noerrorFL
231 9G       move.b #5,Errorstate
232 E4       jsr    changeerror
233 cD0 noerrorFL:
234 GH5      cmp.l  d0,d6
                    ;Neue Blocknummer in
235 DB       bne.s  FLLoop
                    ;Fileheader(Hash)
236 Qe       move   #504,offset
237 FU       jsr    Listeintrag
238 OT       movem.l (a7)+,d0-d7/a0-a6
239 ZF       rts
240 vC0 ;*************** User-Dir versch.
        *****************
241 AA  UDMove:
242 lu5      movem.l d0-d7/a0-a6,-(a7)
243 1o       jsr    HKey
                    ;Neuer Header-Key
244 sg       move.l 500(a0),d0
245 x0       clr.l  d5
246 OB       jsr    SeHashB
                    ;Neue Hashtab.
247 Ab       moveq  #1,d5
248 1N       move.l d6,d0
                    ;Alle Unterdirs
249 Cx       jsr    SeHashB
250 Cf       movem.l (a7)+,d0-d7/a0-a6
251 1R       rts
252 ny0 ;*************** Data-Block vers
        ch.*************
253 iP  DBMove:
254 x65      movem.l d0-d7/a0-a6,-(a7)
255 Ov       move.l ABADR,a0
256 n5       move.l 4(a0),d0
257 SW       move.l 8(a0),d1
                    ;Erster Datenblock?
258 gY       cmp.w  #1,d1
259 np       beq.s  erstblock
260 Gq0 DBLoop1:
261 jv5      jsr    AdrRech
262 d0       adda.l #24,a0
                    ;Blocknummer eintragen
263 sH       moveq  #71,d1
264 Hv0 SHLoop:
265 Ct5      cmp.l  (a0)+,d6
266 1Y       dbeq   d1,SHLoop
267 XM       beq.s  gefunden
                    ;Neue Hashtabelle
268 5y       move.l SBLOCK,a0
269 fD       move.l 504(a0),d0
270 KF       move.l a0,a2
271 g7       bra.s  DBLoop1
272 Vh0 gefunden:
273 d65      move   #-4,offset
274 q5       jsr    Listeintrag
275 Nt       tst.l  (a0)
276 1y       beq.s  Vblock
                    ;Noch NextData eintrage
                    n
277 GB       move.l (a0),d0
278 Qu       bra.s  weiterDB
279 cw0 Vblock:
280 075      move.l 24(a2),d0
281 QT0 weiterDB:
282 4G5      jsr    AdrRech
283 k3       cmp.l  16(a0),d6
284 mo       beq.s  noerrorDB2
                    ;Vorgangerblock
285 6E       move.b #6,Errorstate
286 6w       jsr    changeerror
287 IX0 noerrorDB2:
288 On5      move   #16,offset
289 5K       jsr    Listeintrag
290 s0       bra.s  fertigDB
291 310 erstblock:
292 EQ5      jsr    AdrRech
293 cT       cmp.l  308(a0),d6
294 aU       beq.s  noerrorDB1
295 GO       move.b #6,Errorstate
296 G6       jsr    changeerror
297 Mf0 noerrorDB1:
298 q85      move   #308,offset
                    ;In alle Filelist u.
299 FU       jsr    Listeintrag
300 xY0 DBLoop2:
301 2L5      cmp.l  16(a0),d6
302 cy       beq.s  noerrorDB
303 z2       move.b #1,Errorstate
304 OE       jsr    changeerror
305 qF0 noerrorDB:
306 DY5      move   #16,offset
                    ;Fileheader eintragen
307 Nc       jsr    Listeintrag
308 Iq       move.l 504(a0),d0
309 OT       tst.l  d0
310 bA       beq.s  fertigDB
311 Xj       jsr    AdrRech
312 Lm       bra.s  DBLoop2
313 OU0 fertigDB:
314 Eh5      movem.l (a7)+,d0-d7/a0-a6
315 nT       rts
316 6z0 HKey:
317 Ov5      move.l ABADR,a0
318 6z       cmp.l  4(a0),d6
319 kn       beq.s  weiterHK
                    ;Neuen HeaderKey
320 kt       move.b #7,Errorstate
321 fV       jsr    changeerror
322 r70 weiterHK:
323 QS5      move   #4,offset
324 et       jsr    Listeintrag
325 xd       rts
326 gn0 ;********* SBitMap: Blocknummer i
        n d0 ***********
327 Ni  SBitMap:
328 Kz5      movem.l d0,-(a7)
329 OY       move.l BMADR,d5
330 a2       moveq  #2,d1
331 uH0 SBM1:
332 dC5      addq.l #4,d5
333 qB       add.w  #32,d1
334 Tv       cmp.w  d1,d0
```

```
                    ;Bit in Bitmap testen
         bpl.s  SBM1
         sub.w  d0,d1
         exg.l  d5,a0
         move.l (a0),d5
         sub.w  #32,d1
         neg.w  d1
         btst   d1,d5
         movem.l (a7)+,d0
         rts
BMADR ds.l 1
;********* AdrRech: Blocknummer i
n d0 ************
AdrRech:
         movem.l d0,-(a7)
         cmp.w  #880,d0
         bpl.s  groesserAR
         move.l Chipbuffer,a0
         bra.s  weiterAR
groesserAR:
         move.l Fastbuffer,a0
         sub.w  #880,d0
                    ;Aus Blocknummer Speic
                    her-
weiterAR:
         mulu   #512,d0
                    ;adresse berechnen
         adda.l d0,a0
         move.l a0,SBLOCK
         movem.l (a7)+,d0
         rts
SBLOCK ds.l 1
;***** SeHashB: Bedingung in d5 B
lkn. in d0 *****
SeHashB:
         movem.l d0-d4/a0-a1,-(a7)
         jsr    AdrRech
         adda.l #24,a0
         moveq  #71,d3
SHBLoop:
         move.l (a0)+,d0
         tst.l  d0
         beq.s  weiterSHB1
         jsr    UDeintrag
         cmp.l  #2,d5
         bne.s  keinsort
                    ;Diese Routine durch-
         move.l ABNR,d0
                    ;sucht die Hashtabelle
         bra.s  schonsort
                    ;und fuegt neue Werte e
                    in
keinsort:
         cmp.l  d0,d6
         beq.s  gefundenSHB1
schonsort:
         move.l a0,a1
SHBLoop2:
         jsr    AdrRech
         move.l 496(a0),d0
         tst.l  d0
         beq.s  weiterSHB2
         jsr    UDeintrag
                    ;Such nach HashChain
         cmp.l  #2,d5
         bne.s  keinsort2
         move.l ABNR,d0
         bra.s  SHBLoop2
keinsort2:
         cmp.l  d0,d6
         beq.s  gefundenSHB2
         bra.s  SHBLoop2
weiterSHB2:
         move.l a1,a0
weiterSHB1:
         dbra   d3,SHBLoop
         tst.l  d5
         bne.s  fertigSHB
obenSHB:
         move.b #8,Errorstate
         jsr    changeerror
gefundenSHB1:
         suba.l #500,a0
gefundenSHB2:
         tst.l  d5
         bne.s  obenSHB
         move   #496,offset
         jsr    Listeintrag
fertigSHB:
         cmp.l  #2,d5
         bne.s  fertigSHB2
         jsr    FileSort
fertigSHB2:
         movem.l (a7)+,d0-d4/a0-a1
         rts
UDeintrag:
         tst.l  d5
         beq.s  weiterUDE
         cmp.l  #2,d5
         beq.s  DirSort
         movem.l d0/a0,-(a7)
         jsr    AdrRech
         move.l 500(a0),d0
         cmp.l  d0,d6
                    ;Routine fuer Userdir
         beq.s  noerrorUDE
         move.b #2,Errorstate
         jsr    changeerror
noerrorUDE:
         move   #500,offset
         jsr    Listeintrag
         movem.l (a7)+,d0/a0
weiterUDE:
         rts
DirSort:
         movem.l d0-d2/a0,-(a7)
         jsr    AdrRech
         move.l (a0),d2
         cmp.b  #8,d2
         beq.s  weiterDiSo
         sub.l  508(a0),d2
weiterDiSo:
         move.l d7,d1
         jsr    BlockMove
                    ;Alle Subeintraege!
         addq.l #1,d7
         addq.l #1,d6
         jsr    Newnum
         cmp.l  #5,d2
         bne.s  keinFH
         move.l ABNR,Bbuff
FLSORTLoop:
         move.l ABNR,d0
         jsr    AdrRech
         move.l 504(a0),d0
         tst.l  d0
         beq.s  keinFL
         move.l #19,d2
         move.l d7,d1
                    ;Alle FileList nach
         jsr    BlockMove
                    ;FileHeader eintr.
         addq.l #1,d7
         addq.l #1,d6
         jsr    Newnum
         bra.s  FLSORTLoop
keinFL:
         cmp.b  #19,d2
         beq.s  mehrdata
         move.l 8(a0),d0
         cmp.l  #1,d0
         bne.s  mehrdata
         move.l 16(a0),d0
         move.l d7,d1
                    ;1-Block Data versch.
         move.l #8,d2
         jsr    BlockMove
         addq.l #1,d7
         addq.l #1,d6
         jsr    Newnum
mehrdata:
         move.l Bbuff,ABNR
keinFH:
         movem.l (a7)+,d0-d2/a0
         rts
FileSort:
         move   Firstsub,d0
         ext.l  d0
FSLoopa:
         tst.l  d6
         beq.s  kleinerFHFS
         jsr    AdrRech
         move.l (a0),d1
         cmp.l  #8,d1
         beq.s  keinFHFS
         sub.l  508(a0),d1
         cmp.l  #5,d1
                    ;Alle Datenbloecke
         bne.s  keinFHFS
                    ;des Parentdir
         move.l 8(a0),d1
                    ;suchen...
         cmp.l  #1,d1
         beq.s  keinFHFS
         jsr    Filesortb
keinFHFS:
         addq.l #1,d0
         cmp.l  #1759,d0
         bls.s  kleinerFHFS
         moveq  #2,d0
kleinerFHFS:
         dbra   d6,FSLoopa
         rts
Filesortb:
         movem.l d0-d2,-(a7)
FSBLoop:
         move.l 16(a0),d0
         beq.s  fertigFSB
         move.l d7,d1
         move.l #8,d2
         jsr    BlockMove
                    ;..und sortieren!
         addq.l #1,d7
         jsr    Newnum
         move.l ABNR,d0
         jsr    AdrRech
         bra.s  FSBLoop
fertigFSB:
         movem.l (a7)+,d0-d2
         rts
Newnum:
         cmp.l  #1759,d7
         bls.s  kleinerNN
         moveq  #2,d7
                    ;Bereichstest
kleinerNN:
         rts
Firstsub ds.w 1
Bbuff ds.l 1
;****** ChkSumm: Alle Checksummen
 berechnen ******
ChkSumm:
         moveq  #1,d0
CSLoop:
         addq.w #1,d0
         cmp.w  BitmapBlock,d0
         beq.s  CSBitMap
         cmp.w  #1760,d0
         beq.s  FertigCS
                    ;Berechnen aller Check-
         jsr    SBitMap
                    ;summen ausser Bootbloe
                    cke
         bne.s  CSLoop
         jsr    AdrRech
         clr.l  d4
         moveq  #127,d3
CHKS:
         add.l  (a0)+,d4
         dbra   d3,CHKS
         sub.l  -492(a0),d4
         neg.l  d4
         move.l d4,-492(a0)
         bra.s  CSLoop
FertigCS:
         rts
CSBitMap:
```

Listing 2. (Fortsetzung)

```
        jsr     AdrRech
        moveq   #126,d3
        adda.l  #4,a0
        clr.l   d4
CHKB:
        add.l   (a0)+,d4
        dbra    d3,CHKB
        neg.l   d4
        move.l  d4,-512(a0)
        bra.s   CSLoop
ABNR  ds.l 1
NBNR  ds.l 1
ABADR ds.l 1
NBADR ds.l 1
BitmapBlock ds.w 1
offset ds.w 1
Listadr ds.l 1
Listzaehl ds.w 1
;********* SwapBlock: B.adr. in a
0 u. a1 *********
SwapBlock:
        movem.l d0-d1/a0-a1,-(a7)
        move.l ABADR,a0
        move.l NBADR,a1
        moveq   #127,d0
SBLoop:
        move.l (a0),d1
        move.l (a1),(a0)+
        move.l d1,(a1)+
                ;Bloecke austauschen
        dbra    d0,SBLoop
        movem.l (a7)+,d0-d1/a0-a1
        rts
;*********** SwapBit: B.nr. ABNR
u.NBNR **********
SwapBit:
        movem.l d0-d6/a0-a6,-(a7)
        move.l  ABNR,d0
        jsr     SBitMap
        move.l  a0,a6
        move.l  d1,d4
                ;Belegungsbits vertausc
                hen
        move.l  d5,d6
        move.l  NBNR,d0
        jsr     SBitMap
        cmp.l   d5,d6
        beq.s   gleichSB
        btst    d4,d6
        beq.s   belegt1
        bset    d1,d5
weiter1:
        beq.s   belegt2
        bset    d4,d6
        bra.s   weiter2
belegt1:
        bclr    d1,d5
        bra.s   weiter1
belegt2:
        bclr    d4,d6
weiter2:
        move.l  d6,(a6)
        move.l  d5,(a0)
endeSB:
        movem.l (a7)+,d0-d6/a0-a6
        rts
gleichSB:
        btst    d4,d6
        beq.s   belegt1g
        bset    d1,d6
weiter1g:
        beq.s   belegt2g
        bset    d4,d6
        bra.s   weiter2g
            ;Worte aus Bitmap
belegt1g:
        bclr    d1,d6
            ;gleich!
        bra.s   weiter1g
belegt2g:
        bclr    d4,d6
weiter2g:
        move.l  d6,(a6)
        bra.s   endeSB
Listeintrag:
        movem.l d0/a0-a1,-(a7)
        move    offset,d0
        ext.l   d0
        adda.l  d0,a0
        move    Listzaehl,d0
        ext.l   d0
        move.l  Listadr,a1
                ;Alle Bloecke in Liste
        adda.l  d0,a1
        move.l  a0,(a1)
        addq.l  #4,d0
        move    d0,Listzaehl
        movem.l (a7)+,d0/a0-a1
        rts
Blockeintrag:
        move    Listzaehl,d3
        lsr     #2,d3
        subq    #1,d3
        move.l  Listadr,a0
        move.l  ABNR,d1
BELoop:
        tst.l   (a0)
        beq.s   weiterBE
        move.l  (a0)+,a1
        move.l  d1,(a1)
        dbra    d3,BELoop
loeschen:
        clr     Listzaehl
        move.l  Listadr,a0
            ;Alle Blockaenderungen
        move    #1763,d3
            ;aus Liste  eintragen
clrloop:
        clr.l   (a0)+
        dbra    d3,clrloop
        rts
weiterBE:
        adda.l  #4,a0
        move.l  NBNR,d1
        dbra    d3,BELoop
        bra.s   loeschen
changeerror:
        movem.l d0-d7/a0-a6,-(a7)
        move.l  ABNR,d0
        lea     bnummer,a0
        moveq   #2,d3
            ;Blocknummer ausgeben
        jsr     numtxt
        lea     enummer,a0
            ;Fehlernummer ausgeben
        move.b  Errorstate,d0
        ext.w   d0
        clr.l   d3
        jsr     numtxt
        move.l  ExecBase,a6
        jsr     Permit(a6)
        move.l  #cerrtxt,text
        move.l  #text-cerrtxt,textle
        n       ;schreiben!
        jsr     printtxt
        move.b  Errorstate,d0
        cmp.b   #7,d0
        bls.s   geht
        jsr     ChkSumm
        jsr     swapbuffer
        move.l  SPointer,a7
            ;Hard Error (F.Nr. 8)
        bra     exitchange
geht:
        clr.b   Errorstate
        movem.l (a7)+,d0-d7/a0-a6
        rts
Errorstate ds.b 1
SPointer ds.l 1
;*************** numtxt: Nummer i
n d0 ************
numtxt:
        movem.l d0-d3/a0,-(a7)
        adda.l  d3,a0
        adda.l  #1,a0
nummloop:
        clr     d2
        move    d0,d2
        and     #15,d2
        add     #48,d2
                ;Nummer in Register
        cmp     #'9',d2
                ;d0 ab Adresse a0
        bls     ziffer
              ;mit Laenge-1  in d3
        add     #7,d2
ziffer:
        move.b d2,-(a0)
        lsr     #4,d0
        dbra    d3,nummloop
        movem.l (a7)+,d0-d3/a0
        rts
;******************* Fehlerausga
be ****************************
memfehler:
        move.l #memtxt,text
        move.l #quelltxt-memtxt,tex
        tlen   ;Fehlermeldung
        jsr     printtxt
        jmp     schluss
trackfehler:
        move.l #losatxt,text
               ;Fehlermeldung
        move.l #stxt-losatxt,textle
        n
        jsr     printtxt
        jmp     schluss
;********************* Programme
nde ***************************
ende:
        move.l #endtxt1,text
        move.l #endtxt2-endtxt1,tex
        tlen   ;1.Meldung erstellen
        jsr     printtxt
        clr.b   d0
        jsr     tastloop
        tst.b   d0
              ;L o. R Taste?
        beq     schluss
        move.l #endtxt2,text
        move.l #cerrtxt-endtxt2,tex
        tlen   ;R. Taste!
        jsr     printtxt
tastloop2:
        btst    #10,$dff016
        beq     tastloop2
              ;R. Taste loslassen
        clr.b   d0
        jsr     tastloop
        tst.b   d0
              ;L. Taste:Neue Disk
        beq     weiter
        move.l #clstxt,text
        move.l #memtxt-clstxt,textl
        en      ;R. Taste:Multicopy
        jsr     printtxt
        jsr     swapbuffer
        bra     savedisk
tastloop:
        btst    #6,$bfe001
        beq     links
        btst    #10,$dff016
        bne     tastloop
              ;Tastenauswertung
        add.b   #1,d0
links:
        rts
schluss:
        tst.l   Fastbuffer
        beq     nextbuff
        move.l Fastbuffer,Buffer
        jsr     nomem
              ;Speicher freigeben
nextbuff:
        tst.l   Chipbuffer
        beq     aus
        move.l Chipbuffer,Buffer
        jsr     nomem
aus:
```

```
        rts
weiter:
        jsr     schluss
        jmp     start
taskan:
        move.l  ExecBase,a6
        jsr     Permit(a6)
        tst.b   Errorstate
        bne     ende
;*************** Ganze Diskette zurueckspeichern ***************
savedisk:
        move.l  #zieltxt,text
        move.l  #maustxt-zieltxt,textlen  ;Zieldiskette!
        jsr     printtxt
        jsr     wait
        move.l  #stxt,text
        move.l  #ltxt-stxt,textlen
                ;"saving"ausgeben
        jsr     printtxt
        move    #td_format,Kommando
        move.l  #0,Blk_Nr
        jsr     losablock
                ;Block 0-879 speichern
        jsr     swapbuffer
        move.l  #880,Blk_Nr
        jsr     losablock
                ;Block 880-1759 speichern
        jmp     ende
swapbuffer:
        move.l  Chipbuffer,a1
        move.l  Fastbuffer,a0
        move.l  #112640,d0
transbuff:
        move.l  (a0),d1
                ;Chip <=> Fast
        move.l  (a1),(a0)+
        move.l  d1,(a1)+
        subq.l  #1,d0
        bne.s   transbuff
        rts
Chipbuffer ds.l 1
Fastbuffer ds.l 1
;****************** Warten auf 1. Maustaste ******************
wait:
        move.l  #maustxt,text
        move.l  #losatxt-maustxt,textlen
        jsr     printtxt
waitloop:
        btst    #6,$bfe001
                ;Textausgabe
        bne.s waitloop
                ;und warten
        rts
;******** Speicher: nomem + getmem ******************
;* Routine belegt Speicherbereich           *
;*  Laenge in MemLeange,Speicherart in Bed.  *
;* Rueckgabewerte:                          *
;*       Adresse in Buffer                  *
;****************************************************
getmem:
        move.l  ExecBase,a6
        move.l  MemLaenge,d0
                ;Speicher beantragen
        move.l  Bedingung,d1
        jsr     AllocMem(a6)
        move.l  d0,Buffer
fehler:
        rts
nomem:
        move.l  ExecBase,a6
        move.l  MemLaenge,d0
        move.l  Buffer,a1
                ;Speicher freigeben
        jsr     FreeMem(a6)
        rts
Bedingung ds.l 1
Buffer ds.l 1
MemLaenge ds.l 1
;********* Blockroutine: losablock ******************
;* Uebergabewerte:                           *
;*      Startblocknummer in Blk_Nr           *
;*      Speicher fuer Block(s) in diskbuff   *
;*      Speicherlaenge in bufflen            *
;*      Anweisung in Kommando                *
;* Rueckgabewerte:                           *
;*      Uebertragene Bytes in d6             *
;****************************************************
losablock:
        lea     diskio,a6
        move.l  #27,d0
clearloop:
        move.l  #0,(a6)+
                ;Struktur loeschen
        dbra    d0,clearloop
        move.l  ExecBase,a6
        move.l  Blk_Nr,d0
                ;Blk_Nr mal 512
        mulu    #512,d0
        move.l  d0,Blk_Nr
        sub.l   a1,a1
        jsr     FindTask(a6)
        lea     readreply,a1
                ;Eigenen Task
        move.l  d0,16(a1)
                ;wieder aufwecken
        jsr     AddPort(a6)
        lea     diskio,a1
        move.l  #0,d0
        clr.l   d1
        lea     trddevice,a0
                ;Trackdiskdevice oeffnen
        jsr     OpenDev(a6)
        tst.l   d0
        bne     error
        lea     diskio,a1
        move.l  #readreply,14(a1)
        move    Kommando,28(a1)
                ;Struktur beschreiben
        move.l  diskbuff,40(a1)
                ;und Block laden
        move.l  bufflen,36(a1)
        move.l  Blk_Nr,44(a1)
        move.l  ExecBase,a6
        jsr     DoIo(a6)
        lea     diskio,a1
        move.l  32(a1),d6
        move    #9,28(a1)
                ;Motor ausschalten
        move.l  #0,36(a1)
        jsr     DoIo(a6)
        lea     readreply,a1
        jsr     RemPort(a6)
                ;Port entfernen
        lea     diskio,a1
        jsr     CloseDev(a6)
                ;Trddevice schliessen
error:
        rts
trddevice dc.b 'trackdisk.device',0
diskio ds.l 20
readreply ds.l 8
bufflen ds.l 1
diskbuff ds.l 1
Blk_Nr ds.l 1
Kommando ds.w 1
;************ Textausgabe printtxt: *****************
;*   Textanfang in text                      *
;*   Textlaenge in textlen                   *
;****************************************************
printtxt:
        movem.l d0-d6/a0-a6,-(a7)
        move.l  ExecBase,a6
        lea     dosname,a1
        clr.l   d0
                ;Dos-Lib oeffnen
        jsr     OpenLib(a6)
        beq     falsch2
        move.l  d0,dosbase
        move.l  d0,a6
                ;Outputkanal(Fenster) holen
        jsr     OutPut(a6)
        move.l  d0,d1
        move.l  text,d2
        move.l  textlen,d3
                ;Text ausgeben
        jsr     Write(a6)
        tst.l   d0
        bpl     close
        jmp     Exit(a6)
close:
        move.l  ExecBase,a6
        move.l  dosbase,a1
        jsr     CloseLib(a6)
                ;Dos-Lib schliessen
falsch2:
        movem.l (a7)+,d0-d6/a0-a6
        rts
dosname  dc.b 'dos.library',0
dosbase  ds.l 1
clstxt   dc.b 12,10,10,9,9,155
         dc.b '1;31;42m    FastLoadCopy by P.S.   (C) M&T in 1988    '
         dc.b 155,'0m',10,10,10
memtxt   dc.b 'Zu wenig Speicher vorhanden',10
quelltxt dc.b 'Bitte Quelldiskette in DF0:',10
zieltxt  dc.b 10,'Bitte Zieldiskette in DF0:',10
maustxt  dc.b '... und linke Maustaste druecken!',10
losatxt  dc.b 'Trackdisk.device - Probleme',10
stxt     dc.b '... saving',10,10
ltxt     dc.b '... loading',10,10
ctxt     dc.b '... changing',10,10
endtxt1  dc.b 10,10,10,9,155
         dc.b '0;31;42m  LMaustaste =   Quit      ',155,'0m
         '
         dc.b 155,'0;31;42m  RMaustaste =  Nochmal   ',155,'0m',10
endtxt2  dc.b 10,9,155
         dc.b '0;31;42m  LMaustaste = Diskkopie ',155,'0m
         '
         dc.b 155,'0;31;42m  RMaustaste = Ramkopie   ',155,'0m',10
cerrtxt  dc.b 'Fehler in Block $'
bnummer  dc.b '    . Disk u.U. fehlerhaft !! Fehlernummer '
enummer  dc.b ' ',10
text     ds.l 1
textlen  ds.l 1
```


Listing 2. (Schluß)

Kopieren mit vielen Extras

Jeder Amiga-Fan benötigt ein Programm, das Dateien kopiert. »Supercopy« bietet neben dieser Grundfunktion Features, die bisher nur von kommerziell vertriebenen Programmen, beispielsweise CLI-Mate, bekannt waren. Überzeugen Sie sich selbst von der Leistungsfähigkeit dieses Tools.

Ein Programm zum komfortablen Kopieren einzelner Dateien gehört in jede Diskettensammlung. Aber ein leistungsfähiges Tool muß heute schon mehr bieten, als nur die Funktion »Kopieren«. Löschen und Umbenennen von Files, Anzeigen der Inhalte von Text- und Bilddateien gehören dazu. Auf dem kommerziellen Sektor sind dies beispielsweise »Zing!«, »Diskmaster« und »CLI-Mate«.

Diese Programme bieten viele leistungsfähige Zusatzfunktionen. Ausführliche Tests dieser Tools fanden Sie im Amiga-Magazin, dort sind die Vor- und Nachteile beschrieben. Alle kommerziellen Tools haben jedoch einen gravierenden Nachteil: sie kosten eine Menge Geld.

»Supercopy« ist eine hervorragende Alternative zu den genannten Programmen. Neben der Grundfunktion Kopieren sind folgende Features enthalten:

- Kopieren mit gleichzeitigem Löschen der Quelldatei (move),
- Kopieren ganzer Directories,
- Umbenennen von Dateien (rename),

Komfort und Leistungsvielfalt

- Löschen von Files,
- Erstellen von Directories (makedir),
- Formatieren von Disketten (mit oder ohne Verify),
- Anzeigen von ASCII-Files (type),
- Anzeigen von Binär-Dateien (hex-type),
- Anzeigen der zu kopierenden Blocks,
- Anzeige des freien Speicherplatzes auf einer Disk (Floppy-, Hard- und RAM-Disk),
- automatische Erkennung der angeschlossenen Geräte,
- Eingabe von CLI-Befehlen ohne Öffnen eines neuen Fensters,
- Anzeige von IFF-Files, wenn ein entsprechender Befehl (show) auf der Diskette ist,
- Disk-backup, wenn sich ein Kopierprogramm (diskcopy) im c-Directory befindet.

Viele Funktionen und leichte Bedienung

Trotz der vielen eingebauten Funktionen ist Supercopy sehr leicht zu bedienen. Wenn Sie das Programm zum ersten Mal starten, sollten Sie allerdings nur mit Kopien Ihrer Disketten arbeiten, um eine mögliche Beschädigung, zum Beispiel durch den Befehl »FORMAT«, zu vermeiden.

Das Programm ist in der Sprache C geschrieben. Haben Sie die Listings 1 bis 8 mit dem Checksummer (Seite 159) eingegeben, sind die Programme mit dem Aztec-C V3.6 zu compilieren und anschließend zu linken. Die nötigen Anweisungen lauten:

Compiler-Anweisungen:

Als erstes ist »includes.c« (Listing 1) zu compilieren:

```
cc +H supercopy.pre
includes.c
```

Danach übersetzen Sie die Listings 2 bis 8:

```
sc0.c, sc1.c, sc2.c,
sc4.c, sc5.c und sc6.c:
cc +B -s +I supercopy.pre
-o scx.o scx.c
(X = Name der Datei)
sc3.c:
cc -s +I supercopy.pre
-o sc3.o sc3.c
```

Linken:

```
ln +g +Cd sc0.o sc1.o
sc1.o sc3.o sc4.o sc5.o
sc6.o -o supercopy -lc
```

Das klingt sehr viel komplizierter als es wirklich ist. Auf der Programmservice-Diskette finden Sie das Programm natürlich in lauffähiger Form vor. Hier entfallen lästige Vorarbeiten.

Haben Sie alles richtig gemacht, liegt das Programm als »supercopy« in lauffähiger Form vor. Starten Sie das Programm vom CLI aus mit dem Aufruf

```
supercopy <RETURN>
```

Das Tool wird nun geladen, anschließend erscheint der Arbeitsbildschirm (Bild 1). Jetzt können Sie die gewünschten Manipulationen nach Herzenslust vornehmen.

Die Gliederung des Arbeitsbildschirms

Der Bildschirm zeigt anfangs zwei große leere Fenster links und rechts. Hier erscheinen nach Ihrer Wahl die Inhalte der Speichermedien, zum Beispiel der Diskette in Laufwerk df0:

Die angeschlossenen Geräte sehen Sie im Fenster in der Mitte des Bildschirms. Sollten mehr als vier Geräte vorhanden sein (z.B. drei Floppylaufwerke, das RAM und eine Harddisk), verschieben Sie mit dem vertikalen Balken rechts neben der Anzeige den Fensterausschnitt.

Klicken Sie eine der Bezeichnungen mit der linken Maustaste an, erscheinen im aktivierten Fenster (links oder rechts) die Dateien. »Aktiv« ist immer die Seite (SOURCE oder DESTINATION), deren nach unten zeigende Pfeile sichtbar sind.

Es werden immer die ersten dreizehn Einträge angezeigt. Sollten mehr vorhanden sein, sehen Sie diese nach Verschieben des Balkens rechts neben dem Ausgabefenster (bei gedrückter linker Maustaste nach unten ziehen).

Angezeigt wird anfangs das Haupt-Directory (ROOT). In ein Unterverzeichnis wechseln Sie, indem Sie das entsprechende Verzeichnis mit der rechten Maustaste anklicken.

Als Zusatzinformation sehen Sie im Kasten rechts unter der Dateianzeige den freien Speicherplatz auf dem Datenträger.

Die Wurzeln im Haupt-Directory

Der Inhalt eines zweiten Speichermediums, beispielsweise des RAM, läßt sich auf die gleiche Weise im zweiten Ausgabefenster darstellen. Klicken Sie dazu mit der linken Maustaste »DESTINATION« an und anschließend das gewünschte Speichermedium. Sie können auch hier in ein beliebiges Unterverzeichnis wechseln. Es ist auch möglich, in beiden Fenstern verschiedene Verzeichnisse der gleichen Diskette darzustellen.

Meldungen des Programms

Links neben dem waagerechten Balken unten sehen Sie die Bezeichnung »Message«. Hier erscheinen während der Arbeit mit Supercopy häufig Meldungen des Programms.

Die Schalter in der Mitte des Bildschirms besitzen von oben nach unten folgende Bedeutung:

QUIT: beendet die Arbeit mit Supercopy (Rücksprung ins CLI).

PARENT: zeigt den Inhalt des in dem waagerechten Balken oben dargestellten Verzeichnisses an.

ROOT: Wechselt immer in das Hauptverzeichnis.

ALL: Mit diesem Schalter kennzeichnen Sie alle Dateien des aktiven Verzeichnisses. Diese Funktion nimmt Ihnen die Arbeit ab, alle Files mit der linken Maustaste der Reihe nach anzuklicken. Gekennzeichnete Files erkennen Sie an der reversen Darstellung. Um die Wahl einzelner Files rückgängig zu machen, klicken Sie diese mit der linken Maustaste erneut an.

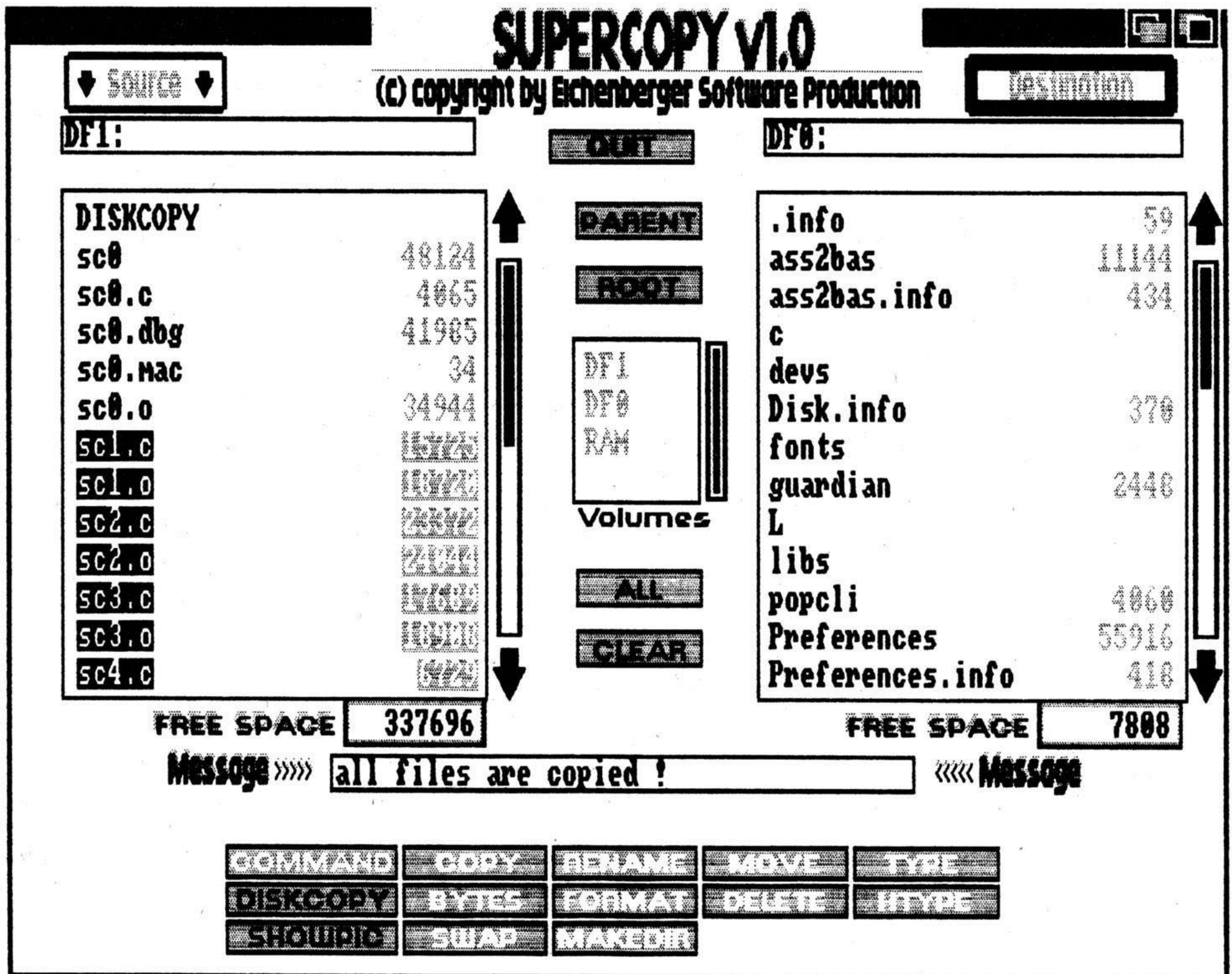


Bild 1. Der Arbeitsbildschirm von »Supercopy« läßt die Funktionsvielfalt des Programms bereits erahnen

CLEAR: hebt die Auswahl und damit die Reversdarstellung aller Dateien im aktiven Verzeichnis auf.

Nachdem wir erfahren haben, welche Möglichkeiten zur Auswahl der Files bestehen, jetzt zu den zahlreichen Funktionen von Supercopy. Im unteren Bildschirmteil erkennen Sie dreizehn Gadgets. Diese tragen beispielsweise die Bezeichnungen »COMMAND« und »COPY«. Folgende Fähigkeiten verbergen sich hinter diesen Schaltern:

COMMAND: Nach Anklicken dieses Schalters mit der linken Maustaste erscheint im Zentrum des Bildschirms ein neues Fenster. Hier sind beliebige CLI-Kommandos einzugeben. Tippen Sie das Kommando (z. B. LIST) ein und bestätigen mit <RETURN>. Statt dessen ist auch ein bereits beim Öffnen des Fensters angezeigter Befehl mit »OK« (linke Maustaste) zu aktivieren. Mit dem Schalter »CANCEL« brechen Sie die Kommando-Eingabe ab.

COPY: Kopieren aller im »Source«-Verzeichnis markierten Dateien in das aktive Verzeichnis im Feld »Destination«.

Spaß durch Vielfalt

Haben Sie Unterverzeichnisse gewählt (diese erscheinen in einer anderen Farbe), werden alle darin enthaltenen Dateien kopiert.

RENAME: Nach Anklicken dieses Schalters erscheint die Aufforderung, eine Datei zu markieren (»please select a File.«). Klicken Sie mit der linken Maustaste eine beliebige Datei an, erscheint ein neues Fenster in der Bildschirmmitte. Der Name des gewählten Files wird angezeigt und ist beliebig zu ändern. Mit <RETURN> oder dem Schalter »OK« bestätigen Sie die Namensänderung, mit »CANCEL« brechen Sie ohne Änderung ab.

MOVE: wie »COPY«. Zusätzlich wird hier nach dem Kopieren die Quelldatei gelöscht.

TYPE: zeigt den Inhalt einer Textdatei an. Nach dem Anklicken ist eine Datei zu wählen. Der Inhalt dieser Textdatei wird auf einem neuen Screen seitenweise ausgegeben. Weiterblättern erfolgt mit <SPACE>. Die Textausgabe ist durch Anklicken des Schließ-Gadgets in der linken oberen Ecke abzubrechen, durch erneutes Anklicken des Gadgets schließen Sie den Screen.

Das Programm erkennt, wenn die gewählte Datei keine Textdatei ist. Die Funktion wird daraufhin abgebrochen mit der Meldung »not correct. Use HTYPE!«.

BYTES: zeigt die Anzahl der Bytes aller gewählten, also revers dargestellten Dateien. Sie können so schnell feststellen, ob alle Files überhaupt auf einen Datenträger passen. Vergleichen Sie einfach die Byte-Zahl mit der Anzeige im Feld »FREE SPACE«.

FORMAT: Formatieren einer neuen Diskette. Es erscheint ein neues Fenster, in dem Sie die Bezeichnung des Laufwerkes (df0:, df1:, df2:) und den Diskettennamen angeben. Wählen Sie zuerst das Laufwerk durch Anklicken der gewünschten Bezeichnung mit der linken Maustaste. Bewegen Sie den Mauszeiger anschließend in das Feld unter der Bezeichnung »DISKNAME« und klicken es an. Geben Sie nun den Namen der zu formatierenden Diskette über die Tastatur ein und drücken <RETURN>. Anschließend startet der Formatiervorgang, die Spuren werden angezeigt.

Wichtig: Supercopy arbeitete im Test beim »Format« fehlerhaft, wenn die Diskettenstation df2: formatiert werden sollte, df1: aber nicht vorhanden war. In diesem Fall steht nur Laufwerk df0: zum Formatieren zur Verfügung.

DELETE: löscht alle selektierten (revers dargestellten) Dateien im aktiven Verzeichnis.

HTYPE: zeigt den Inhalt einer Binär-Datei. Blättern und Abbrechen wie bei »TYPE«.

SWAP: tauscht den Inhalt der Fenster »Source« und »Destination«.

MAKEDIR: legt im aktiven Directory ein neues Unterverzeichnis an. Geben Sie den Namen über die Tastatur ein.

Nützliche Optionen

Supercopy zeigt zwei Schalter in der linken unteren Ecke an. **Die Funktionen »DISKCOPY« und »SHOWPIC« sind nicht im Programm selbst enthalten,** sondern erfordern Zusätze. Dies wurde bewußt auf diesem Wege gelöst, um das Listing in einer vertretbaren Länge zu halten.

DISKCOPY: erfordert im Unterverzeichnis »DISKCOPY« ein Kopierprogramm mit dem gleichen Namen. Auf der Programmservice-Diskette befindet sich allerdings ein Kopierprogramm mit diesem Namen.

SHOWPIC: dient der Anzeige von IFF-Grafiken. Im c-Direktory muß sich zu diesem Zweck ein Programm namens »SHOW« befinden.

Wie schon oben erwähnt sind beide Befehle nicht im Programm enthalten, die Hilfsroutinen sind also vom Anwender eventuell hinzuzufügen.

Zugaben

Mit den Funktionstasten sind noch einige zusätzliche Programmpunkte aufzurufen:

<F1> bis <F3>: Verschiedene Farben sind für den Bildschirm zu wählen.

<F4>: Rückkehr zu den normalen Workbench-Farben.

<F5>: Der Arbeitsbildschirm wird um zirka zehn Prozent verkleinert.

<F6>: Rückkehr zur Originalgröße.

<F7>: Erzeugt ein script-File mit der Bezeichnung »script.asc« im aktuellen Verzeichnis. In dieses File werden alle Directories und Files eingetragen, die selektiert wurden. Bei jedem erneuten Druck auf <F7> werden die gekennzeichneten Files an die bereits bestehende List angehängt. Diese Funktion ist vor allem sinnvoll, wenn Sie eine Übersicht über Ihre Programme und alle Dateien behalten wollen. Mit einem Disksorter können dann diese Files aus der Liste in »script.asc« bequem weiterverarbeitet werden.

Nun sind wir am Ende der Beschreibung aller Befehle des Supercopy angelangt. Wir wünschen Ihnen viel Erfolg beim Austesten der vielen praktischen Funktionen dieses tollen Tools.

(R. Eichenberger/D. Körtke/rs)

Programmname:	includes.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Compiler:	Aztec V3.6a
Bemerkungen:	Bitte beachten Sie die Compiler- und Linker-Anweisungen im Text

```
Programmname: includes.c
------------------------

PeO #include <intuition/intuition.h>
J7  #include <intuition/intuitionbase.h>
60  #include <exec/types.h>
BG  #include <exec/memory.h>
3q  #include <libraries/dos.h>
zx  #include <libraries/dosextens.h>
GR  #include <stdio.h>
(C) 1988 M&T
```

Listing 1. »includes.c« ist als erstes Programm zu compilieren. Bitte geben Sie das Listing mit dem Checksummer (Seite 159) ein.

Programmname:	sc0.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
Programmname: sc0.c
-------------------

NmO /*********************************************************
    ******************
Sc3    Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
       uction
VqO Last UpDate:
SD  Wed Oct  5 14:33:35 1988
Wv  **********************************************************
    *****************/
qF  /*=========================
tU4     HEADERS
IA1  =========================*/
VyO #include<functions.h>
ds  #include<:sc_struct.h>
sD  #include<drawbox.h>
a3  char dummy[1];
BD  struct Source src;
9m  struct Destination dst;
kl  struct Volumes vlm;
jC  struct IntuitionBase *IntuitionBase;
3g  struct GfxBase *GfxBase;
E7  struct Window *Main_Window;
iv  struct Window *Rename_Window;
Np  struct Window *Format_Window;
10  struct RastPort *mainRP;
HN  struct Gadget *sgad[13];  /* SourceBlock Gadgets */
DD  struct Gadget *dgad[13];  /* DestinationBlock Gadgets */
iZ  struct Gadget *vgad[8];   /* VolumesBlock Gadgets */
Ds  #include<vd0:supercopy.h>
WL  main(){
Hn2   OpenLibraries();
6u    InitWindow();
zA3    MemoryAlloc();
gO     SaveDefaultColors();
Wz     Record_Volumes();
kv2   MAIN_WaitMsg();
HgO /*=========================
MB4     Oeffnen der Libraries
jb1  =========================*/
U2O OpenLibraries(){
7P2   if(!(IntuitionBase = (struct IntuitionBase *)OpenLibrary(
      "intuition.library",33L))){
td4     puts("can't open intuition.library");
hB2   if(!(GfxBase       = (struct GfxBase *)OpenLibrary("graph
      ics.library",33L))){
eu4     puts("can't open graphics.library");
PoO /*=========================
uv2   Schliessen der Libraries
rj1  =========================*/
tyO CloseLibraries(){
9f2   CloseLibrary(GfxBase);
1a    CloseLibrary(IntuitionBase);
VuO /*=========================
5v3    Intit. des Hauptwindows
xp1  =========================*/
aQO InitWindow(){
bA2   InitBlockGadgets();
RP    if(!(Main_Window = OpenWindow(&MAIN_window))){
6W4     puts("can't open Mainwindow");
Hd2   mainRP=Main_Window->RPort;
iI    DrawBorder(mainRP,&MAIN_BorderList1,0L,0L);
W5    DrawImage (mainRP,&MAIN_ImageList1 ,0L,0L);
Uc    DrawImage (mainRP,&MAIN_Image36 , 77L,186L);
dB    DrawImage (mainRP,&MAIN_Image36 ,441L,186L);
```

Listing 2. »sc0.c« bitte mit dem Checksummer eingeben

```
    SetAPen(mainRP,1L);
    drawbox (Main_Window->RPort,177L,183L,251L,194L);
    drawbox (Main_Window->RPort,178L,182L,250L,193L);
    SetAPen(mainRP,2L);
    drawbox (Main_Window->RPort,179L,183L,249L,193L);
    SetAPen(mainRP,1L);
    drawbox (Main_Window->RPort,541L,183L,615L,194L);
    drawbox (Main_Window->RPort,542L,182L,615L,193L);
    SetAPen(mainRP,2L);
    drawbox (Main_Window->RPort,543L,183L,614L,193L);
    AddGList (Main_Window,&MAIN_clear,5L,NULL);
    RefreshGList(&MAIN_clear,Main_Window,NULL,5L);
/*=========================
    Init. der BlockGadgets
 =========================*/
InitBlockGadgets(){
  register UBYTE i;
  for(i=0;i<=12;i++){
    sgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
    |MEMF_CHIP);
    CopyGadgets(&MAIN_firstSBlock,sgad[i]);
    sgad[i]->TopEdge = MAIN_firstSBlock.TopEdge+i*10;
    sgad[i]->GadgetID = i+3;
  for(i=0;i<=12;i++){
    dgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
    |MEMF_CHIP);
    CopyGadgets(&MAIN_firstDBlock,dgad[i]);
    dgad[i]->TopEdge = MAIN_firstDBlock.TopEdge+i*10;
    dgad[i]->GadgetID = i+48;
  for(i=0;i<=3;i++){
    vgad[i]=AllocMem((long)sizeof(struct Gadget),MEMF_CLEAR
    |MEMF_CHIP);
    CopyGadgets(&MAIN_firstVBlock,vgad[i]);
    vgad[i]->TopEdge = MAIN_firstVBlock.TopEdge+i*10;
    vgad[i]->GadgetID = i+19;
  for(i=0;i<=11;i++){
     sgad[i]->NextGadget = sgad[i+1];
     dgad[i]->NextGadget = dgad[i+1];
     if(i <= 6) vgad[i]->NextGadget = vgad[i+1];
  sgad[12]->NextGadget  = dgad[0];
  dgad[12]->NextGadget  = vgad[0];
  vgad[7 ]->NextGadget  = NULL;
  MAIN_htype.NextGadget= sgad[0];
MAIN_cleanup(){
   register ULONG i;
  for(i=0;i<=12;i++){
     FreeMem(sgad[i],(long)sizeof(struct Gadget));
     FreeMem(dgad[i],(long)sizeof(struct Gadget));
     if(i <= 3) FreeMem(vgad[i],(long)sizeof(struct Gadge
     t));
   MAIN_htype.NextGadget=NULL;
   MAIN_MAIN_d_pathSInfo.Buffer = (UBYTE *)&dummy;
   MAIN_MAIN_s_pathSInfo.Buffer = (UBYTE *)&dummy;
  if(Main_Window) CloseWindow(Main_Window);
  CloseLibraries();
CopyGadgets(source,destination)
   BYTE *source,*destination;
   register ULONG i;
   for(i=0;i<sizeof(struct Gadget);i++){
      *destination++=*source++;
(C) 1988 M&T
```

Listing 2. (Schluß)

Programmname:	sc1.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
Programmname: sc1.c
-------------------
/*******************************************************
*****************
    Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
    uction
                     Funktionen fuer die Directoryverwaltung
Last UpDate:
Fri Apr 29 20:48:29 1988
********************************************************
************SC1*/
/*=========================
    HEADERS
 =========================*/
#include<exec/types.h>
#include<functions.h>
#include<:sc_struct.h>
/*=========================
    DEFINES
 =========================*/
#define UP 100
#define DOWN 101
#define SOURCE 0L
#define DESTINATION 1L
#define strlen _BUILTIN_strlen
#define strcpy _BUILTIN_strcpy
#define strcmp _BUILTIN_strcmp
/*=========================
    GLOBALE VARIABLEN
 =========================*/
extern LONG Wait_on_Message();
extern struct Source src;
extern struct Destination dst;
extern struct Volumes vlm;
extern struct GfxBase *GfxBase;
extern struct RastPort *mainRP;
extern struct Window *Main_Window;
extern struct StringInfo MAIN_MAIN_s_pathSInfo;
extern struct StringInfo MAIN_MAIN_d_pathSInfo;
extern struct Gadget MAIN_source,MAIN_destination,MAIN_s_pa
th,MAIN_d_path,
               *sgad[13],*dgad[13],*vgad[13];
/*******************************************************
*****************
                 vergleicht 2 Strings (fuer qsort)
********************************************************
****************/
cmp(a,b)
  char **a,**b;
  register int i;
  char str1[180],str2[180],*lcase();
  lcase(str1,*a);
  lcase(str2,*b);
  return(strcmp(str1,str2));
char *lcase(to, from)
  char    *to;
  char    *from;
   char    *temp = to;
   while (*from)
      if (*from < 'A' || *from > 'Z')
          *to++ = *from++;
      else
          *to++ = (*from++ + ' ');
   *to = '\0';
   return(temp);
/*******************************************************
*****************
        Liest das Directory in den Source- oder Destinatio
        n Buffer ein
Sat Apr 30 21:00:46 1988
********************************************************
****************/
Read_Directory(path,flag)
  char path[];
  ULONG flag;
  register int i=0;
  struct FileLock *altlock;
  struct FileLock *neulock; /* Zeiger auf Lock-Structure */
  LONG success;
  neulock = Lock(path,ACCESS_READ);   /* ermittelt LockStru
  cture */
```

Listing 3. »sc1.c« bitte mit dem Checksummer eingeben

```
  if (neulock != 0)
      altlock = CurrentDir(neulock);
      verfolgepfad(neulock,flag,altlock);            /* verfolge Pfad */
      altlock = CurrentDir(altlock);
  else
     WriteMessage("can't lock selected dir");
      return(-1);
   Full_Update(flag);
verfolgepfad(lock,flag,altlock)
struct FileLock *lock;
ULONG flag;
struct FileLock *altlock;
   LONG Wait_on_Message();
  register WORD i;
   struct FileInfoBlock *m;
   struct FileLock *newlock,*oldlock,*ignoredlock;
   LONG gadid;
   WORD success;
   if(!lock) return(0);
   m = (struct FileInfoBlock *)
        AllocMem((long)sizeof(struct FileInfoBlock),MEMF_CLEAR|MEMF_PUBLIC);
   success = Examine(lock,m);
  while (success != 0)
      success = ExNext(lock,m);      /* examine the next entry */
      if(success){
        switch(flag){
        case SOURCE:
                    src.list.entrys++;
                    strcpy(src.list.name[src.list.entrys-1]+i,
                           m->fib_FileName);
                    src.list.length[src.list.entrys-1] = m->fib_Size;
                    if(m->fib_DirEntryType > 0){
                      src.list.dir[src.list.entrys-1] = 1;
                    else
                      src.list.dir[src.list.entrys-1] = 0;
                    break;
        case DESTINATION:
                    dst.list.entrys++;
                    strcpy(dst.list.name[dst.list.entrys-1]+i,
                           m->fib_FileName);
                    dst.list.length[dst.list.entrys-1] = m->fib_Size;
                    if(m->fib_DirEntryType > 0){
                      dst.list.dir[dst.list.entrys-1] = 1;
                    else
                      dst.list.dir[dst.list.entrys-1] = 0;
                    break;
        }/* end of switch */
          }/* end of if */
      }/* end of while */
   if (lock) UnLock(lock);
   FreeMem(m,(long)sizeof(struct FileInfoBlock));
Full_Update(flag)
   LONG flag;
  if((src.list.entrys > 1) && (flag == SOURCE)) Sort_Files(flag);
  if((dst.list.entrys > 1) && (flag == DESTINATION)) Sort_Files(flag);
   Write_Info(flag,2,0);
   DisplayFiles(flag);
   PropGad(flag,0);
   PropGad(flag,1);
/***************************************************************************
                              reserviert Speicher
                          200 Filenamen a 26 Zeichen
                      180 Zeichen fuer den aktuellen Pfad
Sat May 28 14:33:22 1988
***************************************************************************/
MemoryAlloc(){
register ULONG i;
  for(i=0;i<200;i++){
    src.list.name[i] = (char *)AllocMem(27L,MEMF_CLEAR);
  src.spath = (char *)AllocMem(181L,MEMF_CLEAR);
   MAIN_MAIN_s_pathSInfo.Buffer = (UBYTE *)src.spath;
  for(i=0;i<200;i++){
    dst.list.name[i] = (char *)AllocMem(27L,MEMF_CLEAR);
  dst.dpath = (char *)AllocMem(181L,MEMF_CLEAR);
   MAIN_MAIN_d_pathSInfo.Buffer = (UBYTE *)dst.dpath;
  for(i=0;i<30;i++){
    vlm.n[i] = (char *)AllocMem(7L,MEMF_CLEAR);
/***************************************************************************
                 Vorheriges Directory im Pfad aktuell machen
Sun Apr 24 18:16:16 1988
***************************************************************************/
Parent_Dir(pfad,flag)
  char pfad[];
  ULONG flag;
  ULONG count;
  UBYTE depth;
  count=strlen(pfad)-1;
  switch(flag){
    case SOURCE     : depth = src.depth;break;
    case DESTINATION: depth = dst.depth;break;
    default: depth = flag;
   if(depth == 0) return(-1);
  if (depth>1){
    while (pfad[count] != '/'){
       pfad[count]=0;
       count--;
    pfad[count]=0;
  else
    while ((pfad[count] != ':') || (count==0)){
      pfad[count]=0;
      count--;
  depth--;
  switch(flag){
    case SOURCE     : src.depth=depth;break;
    case DESTINATION: dst.depth=depth;break;
    default: break;
/***************************************************************************
               eingegebener Pfad im StringGadget aktuell machen
                                  gibt depth zurueck
Sun Apr 24 19:01:15 1988
***************************************************************************/
UBYTE NewPath(pfad)
  char pfad[];
  register WORD count;
  register UBYTE depth=0;
  count = strlen(pfad);
  while(count > 0){
    while((pfad[--count] != '/') && (count > 0))
    {;}
    depth++;
      if(pfad[strlen(pfad)-1] == ':'){
        depth = 0;
  return(depth);
/***************************************************************************
                 ein neues Directory an einen Pfad anhaengen
Sat Apr 30 21:17:54 1988
***************************************************************************/
PathCat(newpath,flag)
   char newpath[];
   ULONG flag;
   switch(flag){
     case SOURCE:
            if(src.depth > 0) strcat(src.spath,"/");
            src.depth++;
            strcat(src.spath,newpath);
            RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
            break;
     case DESTINATION:
            if(dst.depth > 0) strcat(dst.dpath,"/");
         dst.depth++;
            strcat(dst.dpath,newpath);
            RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
            break;
/***************************************************************************
                  alle reservierten Strukturen zurueckgeben
```

```
n00 Sun Apr 24 20:15:40 1988
vK  ***********************************************************
    *****************/
bX  MEM_cleanup(){
wA3    register ULONG i;
6H2    for(i=0;i<200;i++){
GD4      if(src.list.name[i]) FreeMem(src.list.name[i],27L);
5k2    if(src.spath) FreeMem(src.spath,181L);
9K     for(i=0;i<200;i++){
9K4      if(dst.list.name[i]) FreeMem(dst.list.name[i],27L);
dS2    if(dst.dpath) FreeMem(dst.dpath,181L);
VR     for(i=0;i<30;i++){
Jc4      FreeMem(vlm.n[i],7L);
1Q0 /***********************************************************
    *****************
yAK                         Anzeigen der Files in einer der Bloecke
cJO Sun Apr 24 20:13:39 1988
9Y  ***********************************************************
    *****************/
YY  DisplayFiles(flag)
NB3    ULONG flag;
JR     register UWORD xStart,yStart;
p5     register struct RastPort *rp;
a0     register UWORD i;
HM2   char length[10];
Lx3    register ULONG count;
gM     register char *name;
2p     rp = mainRP;
y8     switch(flag){
Ku6       case SOURCE:
hy9          xStart = 37; yStart=58;
1Q           for(i=0;i<=12;i++){
DwC             ChangeColors();
pR              count = i+src.scroll_count;
ZK              name = src.list.name[count];
CS8          if(src.list.dir[count] == 1) SetAPen(rp,3L);
KF           if(src.list.select[count] == 1){
b3F                SetDrMd(rp,INVERSVID|JAM2);
oHC             if(strlen(name) <= 20){
Y3F                Move(rp,(long)xStart,(long)yStart+i*10);
mP                 Text(rp,name,
oqG                  (long)strlen(name));
jeC             else{
c7F                Move(rp,(long)xStart,(long)yStart+i*10);
qT                 Text(rp,name,
gLG                  20L);
GKC             if(((count) < src.list.entrys) &&
10G                  (src.list.dir[count] == 0)){
f8F                sprintf(length,"%ld",src.list.length[count])
                   ;
EDC             SetAPen(rp,2L);
ZyF                Move(rp,(long)248-TextLength(rp,length,(long
                   )strlen(length)),
k7I                   (long)yStart+i*10);
nUF                Text(rp,length,(long)strlen(length));
CL6       break;
xd        case DESTINATION:
kc9          xStart = 401; yStart=58;
Qp           for(i=0;i<=12;i++){
cLC             ChangeColors();
tY              count = i+dst.scroll_count;
JA9          name = dst.list.name[count];
2G8          if(dst.list.dir[count] == 1) SetAPen(rp,3L);
A3           if(dst.list.select[count] == 1){
0SF                SetDrMd(rp,INVERSVID|JAM2);
DgC             if(strlen(name) <= 20){
xSF                Move(rp,(long)xStart,(long)yStart+i*10);
Bo                 Text(rp,name,
DFG                  (long)strlen(name));
83C             else{
1WF                Move(rp,(long)xStart,(long)yStart+i*10);
Fs                 Text(rp,name,
5kG                  20L);
40C             if(((count) < dst.list.entrys) &&
8YG                  (dst.list.dir[count] == 0)){
mIF                sprintf(length,"%ld",dst.list.length[count])
                   ;
xx                 Move(rp,(long)612-TextLength(rp,length,(long
                   )strlen(length)),
8V                 (long)yStart+i*10);
feC             SetAPen(rp,2L);
CtF                Text(rp,length,(long)strlen(length));
bk6       break;
Fk3    }/* end of switch */
wM0 }/* end of block */
5U  /***********************************************************
    *****************
3aK                    Anzeigen eines Files in einer der Bloec
                       ke
fQ0 Sat Apr 30 15:16:42 1988
Dc  ***********************************************************
    *****************/
ay  DisplaySingle(flag,direction)
RF3    ULONG flag;
CB     ULONG direction;
Rx     register ULONG xStart,yStart;
uA     register struct RastPort *rp;
f5     register UWORD i;
MR2   char length[10];
Q23    register ULONG count;
1R     register char *name;
7u     rp = mainRP;
3D     switch(flag){
Pz6       case SOURCE:
VY9          xStart = 37; yStart=178; i=12;
0C6       if(direction == UP) {i=0;yStart=58;}
tV9          count = i+src.scroll_count;
d0           name = src.list.name[count];
K36       ChangeColors();
HX        if(src.list.dir[count] == 1) SetAPen(rp,3L);
PK        if(src.list.select[count] == 1){
g8C             SetDrMd(rp,INVERSVID|JAM2);
tM9          if(strlen(name) <= 20){
VqC             Move(rp,(long)xStart,(long)yStart);
rU              Text(rp,name,
tvD              (long)strlen(name));
oj9          else{
ZuC             Move(rp,(long)xStart,(long)yStart);
vY              Text(rp,name,
1QD              20L);
LP9          if(((count) < src.list.entrys) &&
6TD              (src.list.dir[count] == 0)){
kDC             sprintf(length,"%ld",src.list.length[count]);
JI9          SetAPen(rp,2L);
e3C             Move(rp,(long)248-TextLength(rp,length,(long)st
                rlen(length)),
w4G                (long)yStart);
sZC             Text(rp,length,(long)strlen(length));
HQ6       break;
2i        case DESTINATION:
KW9          xStart = 401; yStart=178; i=12;
Pb6       if(direction == UP) {i=0;yStart=58;}
xc9          count = i+dst.scroll_count;
nE           name = dst.list.name[count];
JS           ChangeColors();
7L6       if(dst.list.dir[count] == 1) SetAPen(rp,3L);
F8        if(dst.list.select[count] == 1){
5XC             SetDrMd(rp,INVERSVID|JAM2);
I19          if(strlen(name) <= 20){
uFC             Move(rp,(long)xStart,(long)yStart);
Gt              Text(rp,name,
IKD              (long)strlen(name));
D89          else{
yJC             Move(rp,(long)xStart,(long)yStart);
Kx              Text(rp,name,
ApD              20L);
9T9          if(((count) < dst.list.entrys) &&
DdD              (dst.list.dir[count] == 0)){
hg9          SetAPen(rp,2L);
sOC             sprintf(length,"%ld",dst.list.length[count]);
33              Move(rp,(long)612-TextLength(rp,length,(long)st
                rlen(length)),
LTF                (long)yStart);
HyC             Text(rp,length,(long)strlen(length));
gp6       break;
Kp3    }/* end of switch */
1R0 }/* end of block */
Hs  ChangeColors(){
s82   register struct RastPort *rp;
1o3    rp = mainRP;
rp6       SetBPen(rp,0L);
qo        SetAPen(rp,1L);
GV        SetDrMd(rp,JAM2);
Gf0 /***********************************************************
    *****************
```

Listing 3. (Fortsetzung)

```
LwJ                    alle selektierten Filenames-Flags loesch
                       en
qV0 Sat Apr 30 21:26:25 1988
On  ********************************************************
    *****************/
NC  ClearAll(flag)
cQ3    ULONG flag;
Qe     register ULONG i;
7H2   switch(flag){
T36      case SOURCE:
Jb9         for(i=0;i<=199;i++){
3uD            src.list.select[i] = 0;
Jg             *src.list.name[i] = 0;
qG             src.list.length[i] = 0;
ha9         src.list.entrys=0;
4u          src.scroll_count=0;
fJ          SetAPen(mainRP,0L);
jA          EreaseBlock(SOURCE);
6F          break;
rX6      case DESTINATION:
T19         for(i=0;i<=199;i++){
smD            dst.list.select[i] = 0;
Bb             *dst.list.name[i] = 0;
f8             dst.list.length[i] = 0;
WS9         dst.list.entrys=0;
tm          dst.scroll_count=0;
pt          SetAPen(mainRP,0L);
7t          EreaseBlock(DESTINATION);
GP          break;
i70 /*******************************************************
    *****************
roT                               Loescht ein Block-Feld
K80 Sun May 01 13:44:48 1988
qF  ********************************************************
    *****************/
Kc  ClearBlock(flag,count)
4s3    ULONG flag;
8b     UWORD count;
z3     SetAPen(mainRP,0L);
vp     SetDrMd(mainRP,JAM2);
bl     switch(flag){
xX6       case SOURCE:
WyB          RectFill(mainRP,37L,(long)count*10+52,250L,(long
             )count*10+59);
TcC           break;
Eu6       case DESTINATION:
ghB          RectFill(mainRP,401L,(long)count*10+52,614L,(lon
             g)count*10+59);
WfC           break;
Af3    }/* end of switch */
z00 /*******************************************************
    *****************
yGM                        loeschen eines Blockes mit RectFill
n50 Sat May 07 20:16:15 1988
7W  ********************************************************
    *****************/
g2  EreaseBlock(flag)
zd3    LONG flag;
FJ     SetAPen(mainRP,0L);
Xu     SetDrMd(mainRP,0L);
r1     switch(flag){
Dn6       case SOURCE:
ks        RectFill(mainRP,32L,50L,250L,181L);
js9          break;
UA6       case DESTINATION:
f5        RectFill(mainRP,396L,50L,614L,181L);
mv9          break;
Ed0 /*******************************************************
    *****************
no3    Markieren der Files, damit ich nach qsort die alte Pos.
       rausfinde
Kc     danach
M10 ********************************************************
    *****************/
mM  Sort_Files(flag)
a03    ULONG flag;
wT     register int i;
tZ     register UBYTE count;
Fh     char *m;
4g2   UBYTE dir[200];
tr3    ULONG length[200];
9J     switch(flag){
V56       case SOURCE:
9P9          for(i=0;i<src.list.entrys;i++){
9kC             m=src.list.name[i]+strlen(src.list.name[i]);
i3              *(m)=(UBYTE)i+1;
Ov              *(m+1)=(UBYTE)0;
I5              dir[i]=src.list.dir[i];
3o              length[i]=src.list.length[i];
sk9          qsort(src.list.name,(unsigned)(src.list.entrys),(u
             nsigned)4,(ULONG)cmp);
GW           for(i=0;i<src.list.entrys;i++){
GrC             m=src.list.name[i]+strlen(src.list.name[i]);
eS              count = *(m-1);
dzF                src.list.dir[i]   =dir[--count];
kY                 src.list.length[i]=length[count];
md                 *(m-1)=(char)0;
DM8          break;
ye4       case DESTINATION:
3M9          for(i=0;i<dst.list.entrys;i++){
rBC             m=dst.list.name[i]+strlen(dst.list.name[i]);
xI              *(m)=(UBYTE)i+1;
FA              *(m+1)=(UBYTE)0;
I1              dir[i]=dst.list.dir[i];
x1              length[i]=dst.list.length[i];
XV9          qsort(dst.list.name,(unsigned)(dst.list.entrys),(u
             nsigned)4,(ULONG)cmp);
AT           for(i=0;i<dst.list.entrys;i++){
yIC             m=dst.list.name[i]+strlen(dst.list.name[i]);
th              count = *(m-1);
XwF                dst.list.dir[i]   =dir[--count];
eV                 dst.list.length[i]=length[count];
1s                 *(m-1)=(char)0;
Sb8          break;
f00 _cli_parse(){
Lb  _wb_parse(){
(C) 1988 M&T
```

Listing 3. (Schluß)

Programmname:	sc2.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
Programmname: sc2.c
-------------------

NmO /*******************************************************
    *****************
Rb5    Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
       uction
fK6     alle Gadget-Funktionen. z.B. copy, rename, format usw
       .
LqO Tue Apr 26 20:07:35 1988
xu  ********************************************************
    ************SC2*/
qF  /*=======================
tU4    HEADERS
IA1  ========================*/
Vy0 #include<functions.h>
ds  #include<:sc_struct.h>
vK  /*=======================
1v4    DEFINES
NF1  ========================*/
JwO #define SOURCE 0L
v5  #define DESTINATION 1L
OP  /*=======================
X94    GLOBALE VARIABLEN
SK1  ========================*/
zTO extern struct Source src;
7n  extern struct Destination dst;
EO  extern struct Volumes vlm;
DO  extern struct GfxBase *GfxBase;
ph  extern struct RastPort *mainRP;
S1  extern struct Window *Main_Window;
Sk  extern struct DosLibrary *DOSBase;
vU  extern struct NewWindow rename_nw;
q3A        struct Window *Rename_Window;
wr0 extern struct Requester REQ_requester;
GO  extern UBYTE  MessageLine[80];
```

```
wa  extern UBYTE  r_m_buffer[80];
St  extern UBYTE  diskname_buffer[80];
VC  extern struct Gadget MAIN_message;
kC  extern struct Gadget rename_r_m;
VH  extern struct Gadget MAIN_d_path;
FG  extern struct Gadget MAIN_s_path;
3M  extern struct StringInfo MAIN_MAIN_s_pathSInfo;
ae  extern struct StringInfo MAIN_MAIN_d_pathSInfo;
pB  extern struct Image FRM_Image3;
KM  extern struct Image FRM_Image8;
RP  extern struct Image FRM_Image9;
nG  extern struct Image REQ_Image2;
iF  extern struct Image REQ_Image1;
Bs  extern struct Gadget REQ_name;
Vu  extern struct Gadget FRM_cancel;
Gu  extern struct Gadget FRM_ok;
81  extern struct Image command_img;
2k  #define FRM_ImageList1 FRM_Image3
8X  /*************************************************************************
yOI                     kopiert ein File von source auf destination
LGj                                                   0= normal kopieren
JL                                                    1= source loeschen
fC0 Tue Apr 26 19:16:16 1988
Ih  ************************************************************************/
fg  Copy(flag)
Jt3    BOOL flag;
mz     register ULONG i,t=0;
db     struct FileLock *oldlock,*newlock;
kj     char delete[1000];
vP     register LONG entrys;
fH     entrys = src.list.entrys;
67     for(i=0;i<entrys;i++){
xW6       if(src.list.select[i] == 1){
uZ9          if(src.list.dir[i] == 1){
i9C             CopyDir(src.list.name[i],(ULONG)0,(BOOL)flag);
nA              insert_file_entry(i);
CL              t++;
id9          else{
ENC             t++;
Y8              if(CopyFile(src.list.name[i])==-1){
BmA           return(-1);
pAC             if(flag == 1){
yaF                namecat(src.list.name[i],delete,SOURCE);
JF                 if(DeleteFile(delete) == NULL){
DOI                   detect_errors();
gd                    goto end;
e0C             Write_Info(DESTINATION,1,src.list.length[i]);
zM              insert_file_entry(i);
kd0 end:
xI3    if(flag == 1){
PQ6       for(i=0;i<entrys;i++){
dk9          if(src.list.select[i] == 1) kill_file_entry(SOURCE,i--);
C73    WriteMessage("all files are copied !");
bh     ClearAllSelected(SOURCE);
xq     dst.scroll_count=0;
30     if(flag == 1){
J26       print_files(SOURCE);
Yc3    print_files(DESTINATION);
t9     if(t==0){
P26       WriteMessage("first select files (SOURCE).");
nR0 print_files(flag)
6u3    ULONG flag;
D0     EreaseBlock(flag);
ef     Sort_Files(flag);
bk     PropGad(flag,0);
hr     PropGad(flag,1);
Yz     DisplayFiles(flag);
XX     Write_Info(flag,2,0);
2B0 CopyFile(source)
KP3    char source[];
aR     register struct FileHandle *fh1,*fh2;
dF     char filename[500],destination[500];
Q1     char *file_buffer=0;
89     LONG error;
m8     LONG count,count1;
BH     struct FileHandle *Open();
Sm     namecat(source,filename,SOURCE);
5Z     namecat(source,destination,DESTINATION);
YL     WriteMessage(filename);
01     if((fh1 = Open(filename,MODE_OLDFILE))==0){  /* source name */
8H6       puts("can't open");
mm        error = IoErr();
9j        goto cleanup;
Ty3    if((fh2 = Open(destination,MODE_NEWFILE))==0){  /* dst name */
CL6       puts("can't open");
qq        error = IoErr();
Dn        goto cleanup;
S53    file_buffer = (char *)AllocMem(11264L,MEMF_CLEAR);
W3     while(count=Read(fh1,file_buffer,5632L)){
Wh6       if(Write(fh2,file_buffer,(LONG)count) != count){
1T9          WriteMessage("Write-Error");
Vb           if(file_buffer != NULL)   FreeMem(file_buffer,11264L);
JF           Close(fh2);
fA           Close(fh1);
3e           return(-1);
103    if(count == -1){
1I6       WriteMessage("Read-Error");
Qw0 cleanup:
ci3    if(file_buffer != NULL)   FreeMem(file_buffer,11264L);
qM     Close(fh2);
mH     Close(fh1);
GL0 }  /* end of block */
Ut  /*************************************************************************
sxV                                    Requester zeichnen
b00 ************************************************************************/
j4  Draw_request(flag)
cC3    BOOL flag;
GW     register struct RastPort *rp;
kX     register LONG newgadid,success;
Dk     register int i;
Fx     static BOOL execute;
Lv     static struct Gadget ok_cancel[2];
YP     if(execute == 0){
b06       REQ_requester.ReqGadget=NULL;
D3        ok_cancel[0] = FRM_cancel;
Ar        ok_cancel[1] = FRM_ok;
iE        ok_cancel[0].NextGadget = 0;
B7        ok_cancel[0].LeftEdge=5;
8p        ok_cancel[0].TopEdge=35;
aa        ok_cancel[0].Activation |= ENDGADGET;
yM        ok_cancel[0].GadgetType |= REQGADGET;
ji        ok_cancel[0].GadgetID=OK;
Ct        ok_cancel[0].TopEdge=35;
r0        ok_cancel[1].NextGadget = 0;
iI        ok_cancel[1].LeftEdge=139;
ij        ok_cancel[1].Activation |= ENDGADGET;
6V        ok_cancel[1].GadgetType |= REQGADGET;
3m        ok_cancel[1].GadgetID=CANCEL;
K2        ok_cancel[1].TopEdge=35;
Sg        AddGList(Main_Window,&REQ_name,    1L,1L,&REQ_requester);
M8        AddGList(Main_Window,&ok_cancel[0],2L,1L,&REQ_requester);
TH        AddGList(Main_Window,&ok_cancel[1],3L,1L,&REQ_requester);
QJ        execute = 1;
JN3    success = Request(&REQ_requester,Main_Window);
j4     rp = REQ_requester.ReqLayer->rp;
ZA     DrawImage(rp,&FRM_Image8,-1L,-40L);
4Y     DrawImage(rp,&FRM_Image9,-168L,-40L);
fd     SetAPen(rp,1L);
Yp     drawbox(rp,1L,1L,196L,50L);
7H     for(i=0;i<=4;i++){
hY6       Move(rp,9L,(long)(7+i*2));
qR        Draw(rp,185L,(long)(7+i*2));
59        Move(rp,62L,(long)(38+i*2));
V0        Draw(rp,135L,(long)(38+i*2));
p63    if(flag == 0) DrawImage(rp,&REQ_Image2,52L,2L);
zL     if(flag == 1) DrawImage(rp,&REQ_Image1,57L,2L);
XB     if(flag == 2) DrawImage(rp,&command_img,45L,2L);
ZN     Delay(5L);
Zy     success=ActivateGadget(&REQ_name,Main_Window,&REQ_requester);
```

Listing 4. »sc2.c« bitte mit dem Checksummer eingeben

```
/****************************************************************************
                          Files umbenennen
Mon May 02 21:04:45 1988
****************************************************************************/
Call_Rename(flag)
   BOOL flag;
   struct Gadget *igad;
   struct IntuiMessage *Wait_for_Message();
   struct IntuiMessage *Message;
   struct Lock *ignorelock,*dirlock;
   register LONG number,gadid,newgadid;
   LONG Wait_for_Rename();
   char newdir[500];
/*========================
    makedir
 ========================*/
   if(flag == 1){
      makedir();
      return(0);
/*========================
 ========================*/
   WriteMessage("please select a File.");
 FOREVER{
      Message=Wait_for_Message(Main_Window,0);
      gadid = ((struct Gadget *) Message->IAddress)->GadgetID;
      if(Block_Controll(gadid) == 1) continue;
      if( gadid >= 3 && gadid <= 15 ){
         WriteMessage("");
         strcpy(r_m_buffer,src.list.name[gadid-3+src.scroll_count]);
         newgadid=Wait_for_Rename(Message,flag);
         if(newgadid == -1) return(-1);
         RenameFile(src.list.name[gadid-3+src.scroll_count],r_m_buffer,SOURCE);
         strcpy(src.list.name[gadid-3+src.scroll_count],r_m_buffer);
         ClearBlock(SOURCE,(UWORD)gadid-3+src.scroll_count);
         DisplayFiles(SOURCE);
         return();
      if( gadid >= 48 && gadid <= 61 ){
         WriteMessage("");
         strcpy(r_m_buffer,dst.list.name[gadid-48+dst.scroll_count]);
         newgadid=Wait_for_Rename(Message,flag);
         if(newgadid == -1) return(-1);
         RenameFile(dst.list.name[gadid-48+dst.scroll_count],r_m_buffer,DESTINATION);
         strcpy(dst.list.name[gadid-48+dst.scroll_count],r_m_buffer);
         ClearBlock(DESTINATION,(UWORD)gadid-48+dst.scroll_count);
         DisplayFiles(DESTINATION);
         return();
   } /* end of forever */
}/* end of block */
/*========================
 Baue Rename-makedir Req auf
 0 = Rename
 1 = MakeDir
 ========================*/
LONG Wait_for_Rename(Message,flag)
   struct IntuiMessage *Message;
   BOOL flag;
   struct IntuiMessage *Wait_for_Message();
   register LONG newgadid,success;
   register int i;
   Draw_request((BOOL)flag);
   Message=Wait_for_Message(Main_Window,(ULONG)0);
   if(Message == 0) return((long)-1);
   if( (newgadid = ((struct Gadget *) Message->IAddress)->GadgetID) == R_M){
      EndRequest(&REQ_requester,Main_Window);
      return((long)newgadid);
   if((newgadid==OK)){
      EndRequest(&REQ_requester,Main_Window);
      return((long)newgadid);
   else return(-1);
/*========================
    Rename't ein File
 ========================*/
RenameFile(oldname,newname,flag)
   char oldname[],newname[];
   LONG flag;
   char oldfilename[500];
   char newfilename[500];
   switch(flag){
   case SOURCE:
            namecat(oldname,oldfilename,SOURCE);
            namecat(newname,newfilename,SOURCE);
            break;
   case DESTINATION:
            namecat(oldname,oldfilename,DESTINATION);
            namecat(newname,newfilename,DESTINATION);
            break;
   } /* end of switch */
            return(Rename(oldfilename,newfilename));
}/* end of Block */
/****************************************************************************
                          Loeschen eines Files
Sat May 07 14:56:21 1988
****************************************************************************/
Delete(){
   register ULONG i,t=0;
   struct FileLock *oldlock,*newlock;
   char file[500];
   LONG block_type;
   long checkblocks();
   ULONG error;
   block_type = checkblocks();
   switch(block_type){
      case SOURCE:
            for(i=0;i<src.list.entrys;i++){
               if(src.list.select[i] == 1){
                  if(src.list.dir[i] == 1){
            CopyDir(src.list.name[i],1L,0);
                     namecat(src.list.name[i],file,block_type);
                     kill_file_entry(block_type,i);
                  else{
                     t++;
                     WriteMessage(src.list.name[i]);
                     namecat(src.list.name[i],file,block_type);
                     if(DeleteFile(file) == NULL){
                        detect_errors();
                        return(-1);
                     else
                        kill_file_entry(block_type,i);
                  i--;
                  src.scroll_count=0;
            RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
            break;
      case DESTINATION:
            for(i=0;i<dst.list.entrys;i++){
               if(dst.list.select[i] == 1){
                  if(dst.list.dir[i] == 1){
            CopyDir(dst.list.name[i],1L,0);
                     namecat(dst.list.name[i],file,block_type);
                     kill_file_entry(block_type,i);
                  else{
                     t++;
                     WriteMessage(dst.list.name[i]);
                     namecat(dst.list.name[i],file,block_type);
                     if(DeleteFile(file) == NULL){
                        detect_errors();
                        return(-1);
                     else
                        kill_file_entry(block_type,i);
                  i--;
                  src.scroll_count=0;
            RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
            break;
   ClearAllSelected((ULONG)block_type,0);
   src.scroll_count=0;
   dst.scroll_count=0;
   PropGad(block_type,0);
   PropGad(block_type,1);
   DisplayFiles(block_type);
   Write_Info(block_type,2,0);
/**************************************************************
```

```
*****************
                      BCPL-String in C-String umwandeln
Fri Apr 29 20:49:57 1988
******************************************************************************/
BstrC(bstr,buffer)
   BSTR *bstr;
   UBYTE *buffer;
   UBYTE *str;
   LONG loop,zaehler=0;
   str = (UBYTE *) BADDR( bstr );
   for( loop = (LONG) str[0]; loop--; ++zaehler){
      buffer[zaehler] = str[zaehler+1];
   buffer[zaehler] = 0;
/*****************************************************************************
                      Registriere alle Devices im Speicher
Fri Apr 29 20:57:12 1988
******************************************************************************/
Record_Volumes(){
   register ULONG xStart=304,yStart=97,count=0;
   char buffer[10];
   struct RootNode *rootnode;
   struct DosInfo *dosinfo;
   struct DeviceList *devicelist;
   struct FileLock *filelock;
      rootnode = (struct RootNode *) DOSBase->dl_Root;
      dosinfo  = (struct DosInfo *) BADDR( rootnode->rn_Info);
      devicelist = (struct DeviceList *) BADDR( dosinfo->di_DevInfo );
      SetAPen(mainRP,2L);
         while( count <=29 ){
            if( devicelist->dl_Type == 0){  /* nur wenn Device */
               BstrC(devicelist->dl_Name,buffer);
               if( (checkdevice(buffer)) == 0) goto next;
               strcpy(vlm.n[count++],buffer);
               vlm.entrys++;
               if( count <= 4){
                  if( strlen(buffer) <= 6 ){
                     Move(mainRP,(long)xStart,(long)yStart+(count-1)*10);
                     Text(mainRP,vlm.n[count-1],(long)strlen(vlm.n[count-1]) );
            if(devicelist->dl_Next == 0) break;
next:
            devicelist = (struct DeviceList *) BADDR( devicelist->dl_Next );
         PropGad((ULONG)2,(UBYTE)1);
         PropGad((ULONG)2,(UBYTE)0);
/*=========================
    Checke Device
 =========================*/
checkdevice(device)
   char device[];
   static char *devlist[] = {"PRT","SER","CON","RAW","PAR"};
   register LONG i,n;
   for( i=0; i<=4; i++){
      if( (strcmp(device,devlist[i])) == NULL){
         return(0L);
   return(1L);
/*****************************************************************************
                      Schreibt eine Message in die Message-Line
Mon May 02 21:06:22 1988
******************************************************************************/
WriteMessage(text)
   char text[];
   strcpy(MessageLine,text);
   RefreshGList(&MAIN_message,Main_Window,NULL,1L);
/*****************************************************************************
                      Warte auf Message von beliebigem Window
Wed May 04 20:37:43 1988
******************************************************************************/
struct IntuiMessage *Wait_for_Message(MsgWindow,Code)
   struct Window *MsgWindow;
   ULONG Code;
   register ULONG i;
   struct IntuiMessage *Message;
  static struct IntuiMessage Message2;
   BYTE *source,*destination;
  FOREVER
     Wait((LONG)1<<MsgWindow->UserPort->mp_SigBit);
     while ((Message = (struct IntuiMessage *) GetMsg ((struct MsgPort *)MsgWindow->UserPort)))
        source = (BYTE *)Message; destination = (BYTE *)&Message2;
        for(i=0;i<sizeof(struct IntuiMessage);i++){
       *destination++=*source++;
     switch(Code){
        case 0:
           if( (Message->Class == CLOSEWINDOW)){return(0);}
           if( (Message->Class == REQCLEAR)){return(0);}
           if( (Message->Class == GADGETUP) || (Message->Class == GADGETDOWN)
               || (Message->Class == MOUSEBUTTONS)){
          ReplyMsg (Message);
           return((struct IntuiMessage *)&Message2);
           break;
        case 1:
           ReplyMsg(Message);
           return(&Message2);
           break;
/*****************************************************************************
                      File an Pfad und in Zielbuffer
Sat May 07 14:59:47 1988
******************************************************************************/
namecat(pfad,destination,flag)
   char *pfad,*destination;
   LONG flag;
   switch(flag){
      case SOURCE:
         if((strlen(pfad)+strlen(src.spath)) >= 180){
            WriteMessage("Path-Memory full !");
            MAIN_WaitMsg();
         if(src.depth == 0)
            sprintf(destination,"%s%s",src.spath,pfad);
         else
            sprintf(destination,"%s/%s",src.spath,pfad);
         break;
      case DESTINATION:
         if((strlen(pfad)+strlen(dst.dpath)) >= 180){
            WriteMessage("Path-Memory full!");
            MAIN_WaitMsg();
         if(dst.depth == 0)
            sprintf(destination,"%s%s",dst.dpath,pfad);
         else
            sprintf(destination,"%s/%s",dst.dpath,pfad);
         break;
    default:
         if(destination[strlen(destination)-1] == ':'){
            strcat(destination,pfad);
         else{
            strcat(destination,"/");
            strcat(destination,pfad);
/*****************************************************************************
                      alle selektierten Files deselektieren
Sat May 07 16:48:25 1988
******************************************************************************/
ClearAllSelected(flag,kill_dirs)
  ULONG flag,kill_dirs;
  register ULONG i;
 switch(flag){
     case SOURCE:
        for(i=0;i<=src.list.entrys;i++){
           src.list.select[i]=0;
           if(kill_dirs==1) src.list.dir[i]=0;
     break;
     case DESTINATION:
        for(i=0;i<=dst.list.entrys;i++){
           dst.list.select[i]=0;
           if(kill_dirs==1) dst.list.dir[i]=0;
     break;
  EreaseBlock(flag);
```

Listing 4. (Fortsetzung)

```
b2          DisplayFiles(flag);
yN0  /************************************************************************
6eT                                    alle Files selektieren
p10  Sat May 28 13:49:09 1988
6V   ***********************************************************************/
bK   SetAll(flag)
yc3     LONG flag;
8M      register ULONG i;
pz2    switch(flag){
B16       case SOURCE:
Lr9          for(i=0;i<=src.list.entrys;i++){
I8C             src.list.select[i]=1;
ir6       break;
T9        case DESTINATION:
7g9          for(i=0;i<=dst.list.entrys;i++){
1uC             dst.list.select[i]=1;
mv6       break;
b03     EreaseBlock(flag);
tK      DisplayFiles(flag);
Gf0  /************************************************************************
I7M                           loeschen eines Fileeintrags im Block
160  Sat May 07 18:22:55 1988
On   ***********************************************************************/
dl   kill_file_entry(flag,pos)
I13     register LONG flag,pos;
HU      register LONG i;
7H      switch(flag){
T36        case SOURCE:
wM9           for(i=pos;i<src.list.entrys;i++){
LkC              strcpy(src.list.name[i],src.list.name[i+1]);
EA               src.list.select[i]=src.list.select[i+1];
XW               src.list.dir[i]    =src.list.dir[i+1];
kH               src.list.length[i]=src.list.length[i+1];
3d9           src.list.entrys--;
i9            EreaseBlock(SOURCE);
5E            break;
qW6        case DESTINATION:
qw9           for(i=pos;i<dst.list.entrys;i++){
ezC              strcpy(dst.list.name[i],dst.list.name[i+1]);
Rg               dst.list.select[i]=dst.list.select[i+1];
k2               dst.list.dir[i]    =dst.list.dir[i+1];
xn               dst.list.length[i]=dst.list.length[i+1];
rU9           dst.list.entrys--;
5r            EreaseBlock(DESTINATION);
EN            break;
g50  /************************************************************************
EAI                  hineinsetzen eines Filenames in einem Block
w20  Sat May 07 18:40:32 1988
oD   ***********************************************************************/
NV   insert_file_entry(old_position)
E73     ULONG old_position;
PV      register int i=0;
oe      char str1[180],str2[180],*lcase();
by      if(dst.list.entrys > 0){
Ok6        while(i <= dst.list.entrys){
IS9           lcase(str1,dst.list.name[i]);
mt            lcase(str2,src.list.name[old_position]);
im            if(strcmp(str1,str2) == 0)
aBC              return(-1);
KI7         i++;
PB3     strcpy(dst.list.name  [dst.list.entrys],src.list.name  [old_position]);
No      dst.list.dir    [dst.list.entrys]=src.list.dir    [old_position];
yb      dst.list.length[dst.list.entrys]=src.list.length[old_position];
Nq      Sort_Files(DESTINATION);
3c      dst.list.entrys++;
d10  makedir(){
Ba2    struct Gadget *igad;
bq3     struct IntuiMessage *Wait_for_Message();
Qk2    struct IntuiMessage *Message;
kQ3     struct Lock *ignorelock,*dirlock;
Me      register LONG number,gadid,newgadid;
rX      LONG Wait_for_Rename();
CE      char newdir[500];
zi      LONG block_type;
qb      if( (newgadid=Wait_for_Rename(Message,(BOOL)1)) == -1){
rS6        return(-1);
tN3     block_type=checkblocks();
Wt      switch(block_type){
Is6        case SOURCE:
51C              namecat(r_m_buffer,newdir,SOURCE);
aV               strcpy(src.list.name[src.list.entrys],r_m_buffer);
QV               src.list.dir[src.list.entrys]=1;
hD               src.list.select[src.list.entrys]=0;
hD               src.list.entrys++;
s18           break;
dJ4      case DESTINATION:
W6C              namecat(r_m_buffer,newdir,DESTINATION);
Ao               strcpy(dst.list.name[dst.list.entrys],r_m_buffer);
dj               dst.list.dir[dst.list.entrys]=1;
Bn               dst.list.select[dst.list.entrys]=0;
T2               dst.list.entrys++;
z88           break;
X23     if( (ignorelock = Lock(newdir,ACCESS_READ)) == NULL){
fK6        WriteMessage("creating a new directory.");
8U         dirlock=CreateDir(newdir);
iS         UnLock(dirlock);
Ka3     if(ignorelock != NULL) UnLock(ignorelock);
Ts      Sort_Files(block_type);
SS      EreaseBlock(block_type);
15      PropGad(block_type,0);
s8      PropGad(block_type,1);
zz      DisplayFiles(block_type);
b00  /************************************************************************
Uya                                               Command
Hb0  Sat May 28 17:27:16 1988
j8   ***********************************************************************/
o5   command()
qF2    struct Gadget *igad;
GV3     struct IntuiMessage *Wait_for_Message();
oI      char *strchr();
6Q2    struct IntuiMessage *Message;
Q63     struct Lock *ignorelock,*dirlock;
2K      register LONG number,gadid,newgadid;
gN      struct FileHandle *fp;
YE      LONG Wait_for_Rename();
oj2    static char *argv[20];
264      int i=0,t=0,n=0;
dK3     if( (newgadid=Wait_for_Rename(Message,(BOOL)2)) == -1){
Y96        return(-1);
Tg3     Execute(r_m_buffer,0L,0L);
88      WindowToFront(Main_Window);
uJ0  /************************************************************************
HWa                                               SHOWPIC
VN0  Fri Jun 03 19:10:19 1988
2R   ***********************************************************************/
Hx   showpic(){
xM2    static char *argv[50],*execname;
MI3     register int i,n=1;
cK      LONG checkblocks(),totalsize=0,bl_type;
VJ      static UBYTE callname[] = "c:show ";
YB      struct FileLock *lock;
nV2    struct FileLock *altlock;
AM3     argv[0] = callname;
Pe      bl_type = checkblocks();
Mu      switch(bl_type){
Dn6        case SOURCE:
tJ9           lock = Lock(src.spath,ACCESS_READ);
v0            altlock=CurrentDir(lock);
kt            break;
VB4       case DESTINATION:
zq9           lock = Lock(dst.dpath,ACCESS_READ);
zS            altlock=CurrentDir(lock);
ox            break;
V33     switch(bl_type){
Mw6        case SOURCE:
0G9           for(i=0;i<src.list.entrys;i++){
uTC              if(src.list.select[i] == 1){
q1B           argv[n++] = src.list.name[i];
u39           break;
fL6        case DESTINATION:
k39           for(i=0;i<dst.list.entrys;i++){
QxC              if(dst.list.select[i] == 1){
```

```
gHB            argv[n++] = dst.list.name[i];
z89          break;
R53    for(i=1;i<n;i++){
yc6       totalsize+=strlen(argv[i]);
yM3    totalsize+=n+6;  /* fuer Show-Befehl und Spaces */
oL     execname = (char *) AllocMem(totalsize,MEMF_CLEAR);
4z     strcpy(execname,callname);
WA     for(i=1;i<n;i++){
p96       strcat(execname,argv[i]);
de        if(i < (n-1) )strcat(execname," ");
Xs3    Execute(execname,0L,0L);
vU     for(i=0;i<=49;i++){
cr6       argv[i]=0;
QK3    FreeMem(execname,totalsize);
ch     UnLock(lock);
FH     CurrentDir(altlock);
PS     ClearAllSelected(bl_type,0);
H6     DisplayFiles(bl_type);
h60 /*******************************************************
    *****************
1LX                                  ROOT-Directory
UwO Sat May 28 19:19:07 1988
pE  *******************************************************
    *****************/
J5  root_dir(){
OS3    char newdir[80],*strchr();
zI     switch(checkblocks()){
tT9          case SOURCE:
JmC             strcpy(newdir,src.spath);
gi              newdir[strchr(newdir,':')-newdir+1]=0;
f6              strcpy(src.spath,newdir);
fK              src.depth = 0;
Sb              break;
Dt9          case DESTINATION:
b5C             strcpy(newdir,dst.dpath);
mo              newdir[strchr(newdir,':')-newdir+1]=0;
nf              strcpy(dst.dpath,newdir);
Q8              dst.depth = 0;
Yh              break;
PS3     ClearAll(checkblocks());
1I      RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
Bn      RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
Mo      Read_Directory(newdir,checkblocks());
4TO /*******************************************************
    *****************
VJY                                   SWAP-Blocks
uNO Sat May 28 19:19:26 1988
Cb  *******************************************************
    *****************/
mx  swap(){
2G3    struct Source source;
eh     source = src;
dY     src = dst;
Qf     dst = source;
mV     print_files(SOURCE);
15     print_files(DESTINATION);
ov     MAIN_MAIN_s_pathSInfo.Buffer = src.spath;
ZF     MAIN_MAIN_d_pathSInfo.Buffer = dst.dpath;
yY     RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
R3     RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
2r0 detect_errors(){
RS3    LONG error;
yy     error = IoErr();
0b     switch(error){
rh6       case 202:
SG9          WriteMessage("Object is in use !");
y7           break;
1t6       case 213:
9L9          WriteMessage("Disk is not validated.");
1A           break;
zq6       case 221:
Em9          WriteMessage("Disk is full.");
4D           break;
6y6       case 222:
e49          WriteMessage("File is protected from deletion.");
7G           break;
D66       case 223:
369          WriteMessage("File is protected from writing.");
AJ           break;
0J6       case 225:
MH9          WriteMessage("Not a DOS disk.");
DM           break;
VR6       case 226:
BS9          WriteMessage("No disk in drive.");
GP           break;
E46       case 103:
wy9          WriteMessage("insufficient free store.");
JS           break;
714     default:
0D9          WriteMessage("Disk-Error.");
(C) 1988 M&T
```

Listing 4. (Schluß)

Programmname:	sc3.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
NmO /*******************************************************
    *****************
Sc6       Supercopy v1.0 (w) in 1988 by Eichenberger Software P
          roduction
HbR                         Window-Aktionen Funktionen
3GO Tue Apr 26 20:20:41 1988
3w  *******************************************************
    ************SC3*/
qF  /*========================
tU4    HEADERS
IA1  ========================*/
Vy0  #include<functions.h>
ds   #include<:sc_struct.h>
vK  /*========================
1v4    DEFINES
NF1  ========================*/
Jw0  #define SOURCE 0L
v5   #define DESTINATION 1L
JK   #define UP 100
bL   #define DOWN 101
2R  /*========================
ZB4    GLOBALE VARIABLEN
UM1  ========================*/
JL0 struct Source src;
Hu  struct Destination dst;
st  struct Volumes vlm;
7J  LONG Wait_on_Message();
GR  extern struct GfxBase *GfxBase;
sk  extern struct RastPort *mainRP;
V4  extern struct Window *Main_Window;
gZ  extern struct Gadget MAIN_source,MAIN_destination,MAIN_s_pa
    th,MAIN_d_path,
BiL                      MAIN_s_prop,MAIN_d_prop,MAIN_dev_prop;
Ou0 extern struct PropInfo MAIN_MAIN_s_propSInfo,MAIN_MAIN_d_pr
    opSInfo,MAIN_MAIN_dev_propSInfo;
bg  USHORT oldcolor[5];
sH  /*******************************************************
    *****************
EWV                                 Warte auf Message
vt0 Sun May 01 16:16:23 1988
0P  *******************************************************
    *****************/
03  MAIN_WaitMsg(){
xp2   register ULONG Class;
r2    register USHORT Code,Qualifier;
Gt3    register SHORT MouseX,MouseY;
NO2   register struct IntuiMessage *Message;
DY3    register struct Window *IDCMPwindow;
Ag2   int i;
Fw    FOREVER
086        Wait((LONG)1<<Main_Window->UserPort->mp_SigBit);
x95       while ((Message = (struct IntuiMessage *) GetMsg ((str
          uct MsgPort *)Main_Window->UserPort)))
Sq7         Class = Message->Class;
HB8          Code  = Message->Code;
tD9           MouseX= Message->MouseX;
0M            MouseY= Message->MouseY;
07            Qualifier= Message->Qualifier;
gr6        IDCMPwindow = Message->IDCMPWindow;
2a8          ReplyMsg (Message);
jd6        switch (Class){
```

Listing 5. »sc3.c« bitte mit dem Checksummer eingeben

```
OSC                case RAWKEY:
JW                 Key_Call(Code,Qualifier);
pyI                        break;
LfC                case ACTIVEWINDOW:
L2I                        MoveWindow(Main_Window,0L,(long)-(Main_Wi
                           ndow->TopEdge));
w0                         SizeWindow(Main_Window,0L,(long)256-(Main
                           _Window->Height));
t2                         break;
rF8            case MOUSEBUTTONS :
BQI                        if(Code != MENUDOWN) break;
Zg                         Call_Mouse(MouseX,MouseY,Code);
x6T                                  break;
2t9             case REQCLEAR      :
z8S                                 break;
Z09             case GADGETDOWN    : do_gadgets(Message);
1AS                                 break;
bPA              case GADGETUP     : do_gadgets(Message);
3CS                                 break;
fm8            } /* End of switch */
Sj             Class = 0; Code = 0;
Wq4        }  /* End of while */
Yt2     } /* End of forever */
m90 } /* End of Wait_Msg */
az  /*******************************************************
    *****************
hvT                               Gadgets kontrollieren
HNO Sun May 01 16:15:59 1988
i7  *******************************************************
    ****************/
ks1  do_gadgets(mes)   /* Frage welches Gadget gedrueckt wurde,
      danach die
voQ                              gewaehlten Commands ausfuehren ..
                                                 */
1j1  struct IntuiMessage *mes;
rG3    struct Gadget *igad;
5E     BOOL Block_Controll(),success;
VS     register LONG gadid,i,number,number2,number3;
9r     igad = (struct Gadget *) mes->IAddress;  /* Ptr auf Gad
       get */
ax     gadid = igad->GadgetID; /* Eigene Identitaetsnummer */
li     number = (gadid-3)+src.scroll_count;
ns     number2 = (gadid-19)+vlm.scroll_count;
7B     number3 = (gadid-48)+dst.scroll_count;
Dc0 /*========================
pa4     SOURCE-Block
fX1  =========================*/
Nz3    if( (gadid >= 3 && gadid <= 15) && (number <= src.lis
       t.entrys-1) ){
4K8        if(src.list.select[ number ] == 0)
FW7       src.list.select[ number ]=1;
OB5     else
EUA          src.list.select[ number ]=0;
kH7       ClearBlock(SOURCE,gadid-3);
60        DisplayFiles(SOURCE);
Nm0 /*========================
D44     DESTINATION-Block
ph1  =========================*/
7S3    if( (gadid >= 48 && gadid <= 61) && (number3 <= dst.l
       ist.entrys-1) ){
Z18        if(dst.list.select[ number3 ] == 0)
q07       dst.list.select[ number3 ]=1;
YL5     else
oLA          dst.list.select[ number3 ]=0;
gR7       ClearBlock(DESTINATION,gadid-48);
pF        DisplayFiles(DESTINATION);
Xw0 /*========================
We4     VOLUMES-Block
zr1  =========================*/
5y3    if( (gadid >= 19 && gadid <= 27) && (number2 <= vlm.e
       ntrys-1) ){
1K9          switch(checkblocks() ){
JtC             case SOURCE:
WqI                     src.depth=0;
UN                      src.list.entrys=0;
VP                      strcpy(src.spath,vlm.n[ number2 ]);
8b                      strcat(src.spath,":");
S4                      RefreshGList(&MAIN_s_path,Main_Window,NUL
                        L,1L);
wb                      ClearAll(SOURCE);
tU                      Read_Directory(src.spath,SOURCE);
v4                      break;
gMC             case DESTINATION:
KhI                     dst.depth=0;
IE                      dst.list.entrys=0;
g1                      strcpy(dst.dpath,vlm.n[ number2 ]);
JD                      strcat(dst.dpath,":");
9j                      RefreshGList(&MAIN_d_path,Main_Window,NUL
                        L,1L);
Ay                      ClearAll(DESTINATION);
TB                      Read_Directory(dst.dpath,DESTINATION);
4D                      break;
iD6        }/* end of switch */
e13    } /* end of if */
GD     success=Block_Controll(gadid);
Im     if(success == 1) return();
lh     switch(gadid){
509          case COPY:
vYF                Copy();
CL                 break;
Mj9          case RENAME:
4wF                Call_Rename((BOOL)0);
FO                 break;
pY6         case DELETE:
pGF                Delete();
IR                 break;
619          case FORMAT:
51F                InitFormatReq();
LU                 break;
hf9          case COMMAND:
Z6A           command();
0XF                break;
gI9          case BYTES:
pfF                call_bytes();
Ra                 break;
TK9          case MOVE:
TsF                Copy((BOOL)1);
Ud                 break;
WD9          case MAKEDIR:
RKF                Call_Rename((BOOL)1);
Xg                 break;
tv9          case TYPE:
jTF                Type();
aj                 break;
gT9          case HTYPE:
oHF                HType();
dm                 break;
mh9          case SWAP:
foF                break;
tq9          case DISKCOPY:
5nF                fexecl("c:diskcopy","c:diskcopy");
ir                 break;
Qf9          case SHOWPIC:
I6F                showpic();
lu                 break;
XP3    }  /* end of switch */
bt0 }  /* end of block */
Fe  /*******************************************************
    *****************
xHA          untersucht, welcher Block aktiv ist SOURCE oder D
             ESTINATION
yh0 Sat Apr 30 21:10:58 1988
Nm  *******************************************************
    ****************/
Gk  checkblocks(){
YJ2   if( MAIN_source.Flags & SELECTED) return(SOURCE);
I6    if( MAIN_destination.Flags & SELECTED) return(DESTINATION
      );
M10 /*******************************************************
    *****************
QZ5      Wenn rechte Maustaste ueber Directory gedrueckt wurde,
          aktuell machen
Ph0 Sat May 07 13:10:37 1988
Ut  *******************************************************
    ****************/
08  Call_Mouse(x,y,Code)
Fg3    SHORT x,y;  /* Mauskoordinaten */
ub     USHORT Code;
5XC             if( (x >= 37) && (x <= 242) && (y <= 179) ){
VMF                if(src.list.dir[((y-52)/10) + src.scroll_cou
                     nt] == 1){
HWI                   PathCat(src.list.name[((y-52)/10) + src.s
                        croll_count],SOURCE);
8n                    ClearAll(SOURCE);
5g                    Read_Directory(src.spath,SOURCE);
nrC             if( (x >= 401) && (x <= 506) && (y <= 179) )
                {
gYF                if(dst.list.dir[((y-52)/10) + dst.scroll_cou
```

```
                    nt] == 1){
                        PathCat(dst.list.name[((y-52)/10) + dst.scroll_count],DESTINATION);
                    ClearAll(DESTINATION);
                        Read_Directory(dst.dpath,DESTINATION);
}/* end of block */
/***********************************************************************
                                Errechnet bei:  what = 0 -> neue Position
                                                what = 1 -> neue Groesse
Sun May 01 16:15:40 1988
***********************************************************************/
PropGad(flag,what)
   ULONG flag;
   UBYTE what;
   switch (flag) {
      case SOURCE:
      switch(what){
           case 0:
               if(src.list.entrys >= 14){
           MAIN_MAIN_s_propSInfo.VertPot=((ULONG)src.scroll_count*65535/((src.list.entrys-1)-12));
               else
           MAIN_MAIN_s_propSInfo.VertPot=0;
               break;
           case 1:
               if(src.list.entrys >= 14){
                  MAIN_MAIN_s_propSInfo.VertBody = (ULONG)65535/(src.list.entrys/13+1);
               else
                  MAIN_MAIN_s_propSInfo.VertBody = 65535;
        break;
         RefreshGList(&MAIN_s_prop,Main_Window,NULL,1L);
         break;
      case DESTINATION:
      switch(what){
           case 0:
               if(dst.list.entrys >= 14){
           MAIN_MAIN_d_propSInfo.VertPot=((ULONG)dst.scroll_count*65535/((dst.list.entrys-1)-12));
               else
           MAIN_MAIN_d_propSInfo.VertPot=0;
               break;
           case 1:
               if(dst.list.entrys >= 14){
                  MAIN_MAIN_d_propSInfo.VertBody = (ULONG)65535/((dst.list.entrys/13+1));
               else
                  MAIN_MAIN_d_propSInfo.VertBody = (ULONG)65535;
        break;
         RefreshGList(&MAIN_d_prop,Main_Window,NULL,1L);
         break;
      case 2:  /* ProportionalGadget fuer Devices und Volumes */
      switch(what){
           case 0:
                 if(vlm.entrys >= 5)
           MAIN_MAIN_dev_propSInfo.VertPot=((ULONG)vlm.scroll_count*65535/((vlm.entrys-1)-3));
                 else
           MAIN_MAIN_dev_propSInfo.VertPot=0;
              break;
           case 1:
               if(vlm.entrys >= 5){
                  MAIN_MAIN_dev_propSInfo.VertBody = 65535/(vlm.entrys/4+1);
               else
                  MAIN_MAIN_dev_propSInfo.VertBody = 65535;
        break;
         RefreshGList(&MAIN_dev_prop,Main_Window,NULL,1L);
         break;
   } /* end of switch */
/***********************************************************************
                                    Auswertung von RAWKEYCodes
Mon May 02 18:48:31 1988
***********************************************************************/
Key_Call(Code,Qualifier)
   USHORT Code,Qualifier;
   register struct ViewPort *vp;
   static BOOL resizing;
   vp = &(Main_Window->WScreen->ViewPort);
  switch(Code){
     case 0x50:
        SetRGB4(vp,0L,0x5L,0x5L,0x5L);
        SetRGB4(vp,1L,0xdL,0xbL,0x9L);
        SetRGB4(vp,2L,0x0L,0x0L,0x0L);
        SetRGB4(vp,3L,0xfL,0x7L,0x0L);
        break;
     case 0x51:
        SetRGB4(vp,0L,0xaL,0xaL,0xaL);
        SetRGB4(vp,1L,0x2L,0x4L,0x6L);
        SetRGB4(vp,2L,0xfL,0xfL,0xfL);
        SetRGB4(vp,3L,0x0L,0x8L,0xfL);
        break;
     case 0x52:
        SetRGB4(vp,0L,0x0L,0x5L,0xaL);
        SetRGB4(vp,1L,0xfL,0xfL,0xfL);
        SetRGB4(vp,2L,0xcL,0xcL,0xfL);
        SetRGB4(vp,3L,0x0L,0x8L,0xfL);
        break;
     case 0x53:
        Recall_default_color();
        break;
     case 0x54:
        if(resizing == 0){
           SizeWindow(Main_Window,(long)0,(long)-47);
           MoveWindow(Main_Window,(long)0,(long)20);
           resizing=1;
        break;
     case 0x55:
        if(resizing == 1){
           MoveWindow(Main_Window,0L,(long)-(Main_Window->TopEdge));
           SizeWindow(Main_Window,0L,(long)256-(Main_Window->Height));
           resizing=0;
        break;
     case 0x56:
     write_turbo_script();
        break;
/***********************************************************************
                         Sichert die aktuellen Workbench-Farben
Mon May 02 19:45:10 1988
***********************************************************************/
SaveDefaultColors(){
   register BYTE i;
   for(i=0;i<=3;i++){
      oldcolor[i] = GetRGB4(Main_Window->WScreen->ViewPort.ColorMap,(long)i);
/***********************************************************************
               Schreibt die Default-Farben wieder in die ColorMap ein.
Mon May 02 20:21:08 1988
***********************************************************************/
Recall_default_color(){
   register BYTE i;
   register UWORD *ColorTable;
   LoadRGB4(&(Main_Window->WScreen->ViewPort),oldcolor,4L);
/***********************************************************************
                 Rechnet Anzahl Bytes, die kopiert werden sollen, aus.
Sun Apr 24 17:13:28 1988
***********************************************************************/
call_bytes(){
   register int i;
   register ULONG total=0;
   char text[10];
   switch(checkblocks()){
      case SOURCE:
         for(i=0;i<src.list.entrys;i++){
            if(src.list.select[i] == 1){
               total+=src.list.length[i];
         break;
      case DESTINATION:
```

Listing 5. (Fortsetzung)

```
339 CV9          for(i=0;i<dst.list.entrys;i++){
340 sPC             if(dst.list.select[i] == 1){
341 5oF                total+=dst.list.length[i];
342 Ra9          break;
343 q13    sprintf(text,"%ld",total);
344 q62   WriteMessage(text);
345 vKO /*****************************************************
    *****************
346 sY3    totales CleanUp fuer Debugger
347 2RO *****************************************************
    *****************/
348 Na  _abort(){
349 eE3    puts("you have QUIT me !!!!");
350 aU     MEM_cleanup();
351 iw     MAIN_cleanup();
352 fA0 exit(TRUE);
353 3S  /*****************************************************
    *****************
354 8pS                              Fuer die Block-Kontrolle
355 QD0 Sun May 22 13:20:09 1988
356 Ba  *****************************************************
    *****************/
357 Lr  BOOL Block_Controll(gadid)
358 k63    LONG gadid;
359 1I     register int i;
360 r7     register struct RastPort *rp;
361 NW     register ULONG xStart=304,yStart=97,count=0;
362 Qj     BOOL success;
363 2p     rp = mainRP;
364 PL     switch(gadid){
365 Ku9          case SOURCE:
366 gSF                if( (MAIN_source.Flags & SELECTED) == 0){
367 JCI                   MAIN_destination.Flags = GADGHIMAGE+GADGI
                      MAGE+SELECTED;
368 2K                    RefreshGList(&MAIN_destination,Main_Windo
                      w,NULL,1L);
369 aVF                else{
370 LuI                   MAIN_destination.Flags = GADGHIMAGE+GADGI
                      MAGE;
371 5N                    RefreshGList(&MAIN_destination,Main_Windo
                      w,NULL,1L);
372 kqF                success=1;
373 w5                 break;
374 hN9          case DESTINATION:
375 NFF                if( (MAIN_destination.Flags & SELECTED) == 0
                   ){
376 uhI                   MAIN_source.Flags = GADGHIMAGE+GADGIMAGE+
                      SELECTED;
377 mv                    RefreshGList(&MAIN_source,Main_Window,NUL
                      L,1L);
378 jeF                else{
379 wPI                   MAIN_source.Flags = GADGHIMAGE+GADGIMAGE;
380 py                    RefreshGList(&MAIN_source,Main_Window,NUL
                      L,1L);
381 tzF                success=1;
382 5EA          break;
383 Rs6       case S_PATH:
384 ApF                ClearAll(SOURCE);
385 cyA          src.depth = NewPath(src.spath);
386 13F                if( (strncmp(src.spath,"ram:",4)) == NULL){
387 ZXI                   strcpy(src.spath,"RAM:");
388 A1F                Read_Directory(src.spath,SOURCE);
389 17                 success=1;
390 DM                 break;
391 5H6       case D_PATH:
392 NBF                ClearAll(DESTINATION);
393 c2A          dst.depth = NewPath(dst.dpath);
394 cNF                if( (strncmp(dst.dpath,"ram:",4)) == NULL){
395 j8I                   strcpy(dst.dpath,"RAM:");
396 JRF                Read_Directory(dst.dpath,DESTINATION);
397 9F                 success=1;
398 LU                 break;
399 8C9          case SPROP:
400 rmF                src.scroll_count = ((ULONG)(src.list.entrys-
                   13)*(ULONG)MAIN_MAIN_s_propSInfo.VertPot)/0xf
                   fff;
401 z3A          SetAPen(mainRP,0L);
402 RZF                RectFill(mainRP,32L,50L,250L,181L);
403 zt                 DisplayFiles(SOURCE);
404 GM                 success=1;
405 SbA          break;
406 1a9          case DPROP:
407 aHF                dst.scroll_count = ((ULONG)(dst.list.entrys-
                   13)*(ULONG)MAIN_MAIN_d_propSInfo.VertPot)/0xf
                   fff;
408 6AA          SetAPen(mainRP,0L);
409 QqF                RectFill(mainRP,396L,50L,614L,181L);
410 f5                 DisplayFiles(DESTINATION);
411 NT                 success=1;
412 ZiA          break;
413 Yg9        case DEVPROP:
414 geC           SetBPen(rp,0L);
415 ih            SetAPen(rp,2L);
416 5K            SetDrMd(rp,JAM2);
417 9tF               vlm.scroll_count = ((ULONG)(vlm.entrys-4)*(U
                  LONG)MAIN_MAIN_dev_propSInfo.VertPot)/0xffff;
418 BM                for(i=1;i<=4;i++){
419 6eI                  if( strlen() <= 6 ){
420 i5L                     Move(mainRP,(long)xStart,(long)yStart+
                        (i-1)*10);
421 No                      Text(mainRP,vlm.n[i-1+vlm.scroll_count
                        ],(long)strlen(vlm.n[i-1+vlm.scroll_cou
                        nt]) );
422 jsF               break;
423 Yb7      case QUIT:
424 mgF               MEM_cleanup();
425 u8                MAIN_cleanup();
426 rM                exit(TRUE);
427 dj                success=1;
428 py                break;
429 8J9        case CLEAR:
430 1HF               ClearAllSelected((ULONG)checkblocks());
431 s1A        break;
432 KH9        case ALL:
433 JRF               SetAll((LONG)checkblocks());
434 v4                break;
435 e66    case SUP:
436 v1F               do{
437 AmE             if(src.scroll_count > 0){
438 CxL                     ScrollRaster(mainRP,0L,-10L,37L,52L,24
                        9L,179L);
439 3aF               src.scroll_count--;
440 B1G                DisplaySingle(SOURCE,100L);
441 kqL                   PropGad(SOURCE,0);
442 hSI                 Delay(2L);
443 6X9      }while(!(64 & *((UBYTE *)0xbfe001)));
444 uOF               success=1;
445 6F                break;
446 Ze9        case SDOWN:
447 6wF               do{
448 NdI                  if((src.list.entrys >= 14) && (src.scrol
                     l_count <= (src.list.entrys)-14)){
449 1wL                     ScrollRaster(mainRP,0L,10L,37L,52L,249
                        L,179L);
450 OTF               src.scroll_count++;
451 R2G                DisplaySingle(SOURCE,101L);
452 v1L                   PropGad(SOURCE,0);
453 sdI                 Delay(2L);
454 HiA       }while(!(64 & *((UBYTE *)0xbfe001)));
455 5BF               success=1;
456 HQ                break;
457 LL6    case DSTUP:
458 H7F               do{
459 xXE             if(dst.scroll_count > 0){
460 z3L                     ScrollRaster(mainRP,0L,-10L,401L,52L,6
                        13L,179L);
461 4eF               dst.scroll_count--;
462 cLG                DisplaySingle(DESTINATION,100L);
463 fHL                   PropGad(DESTINATION,0);
464 3oI                 Delay(2L);
465 St9      }while(!(64 & *((UBYTE *)0xbfe001)));
466 GMF               success=1;
467 Sb                break;
468 tJ9        case DSTDOWN:
469 SIF               do{
470 bqI                  if((dst.list.entrys >= 14) && (dst.scrol
                     l_count <= (dst.list.entrys)-14)){
471 WjL                     ScrollRaster(mainRP,0L,10L,401L,52L,61
                        3L,179L);
472 1XF               dst.scroll_count++;
473 scG                DisplaySingle(DESTINATION,101L);
474 qSL                   PropGad(DESTINATION,0);
475 EzI                 Delay(2L);
476 d4A       }while(!(64 & *((UBYTE *)0xbfe001)));
477 RXF               success=1;
478 dm                break;
479 hH9        case PARENT:
480 yMA         switch( checkblocks() ){
481 CmI                case SOURCE:
482 GpO                      Parent_Dir(src.spath,SOURCE);
```

```
IuG                 RefreshGList(&MAIN_s_path,Main_Window,NULL,1L);
mR                  ClearAll(SOURCE);
JKO                         Read_Directory(src.spath,SOURCE);
lu                          break;
WCC             case DESTINATION:
uBO                         Parent_Dir(dst.dpath,DESTINATION);
wWG                 RefreshGList(&MAIN_d_path,Main_Window,NULL,1L);
xl                  ClearAll(DESTINATION);
GyO                         Read_Directory(dst.dpath,DESTINATION);
r0                          break;
hnF                success=1;
t2                 break;
qu6        case ROOT:
qaA            root_dir();
w5F                break;
509            case SWAP:
x6F                swap();
z8                 break;
FH9            default: success=0;break;
993        }/*end of switch*/
CZ2    return(success);
o10 }/* end of Block */
Vu  /*********************************************************************
                     Testet ob Message vorliegt, wenn ja -> TRUE
460 Wed Jun 01 11:07:46 1988
d2  *********************************************************************/
tm  LONG Wait_on_Message(window)
V13    struct Window *window;
OX     register struct IntuiMessage *message;
Ea     LONG gadid;
pV     message = (struct IntuiMessage *)window->UserPort->mp_MsgList.lh_Head;
uS     while (message->ExecMessage.mn_Node.ln_Succ)
vQ     gadid = (struct Gadget *) (message->IAddress)->GadgetID;
Cz6        if (message->Class == GADGETUP) {
Ea9            message=GetMsg(window->UserPort);
TU             ReplyMsg(message);
HC             return(gadid);
Y16        if (message->Class == GADGETDOWN){
Ie9            message=GetMsg(window->UserPort);
XY             ReplyMsg(message);
LG             return(gadid);
Sr6        message = (struct IntuiMessage *)message->ExecMessage.mn_Node.ln_Succ;
Zq3     return (FALSE);
(C) 1988 M&T
```

Listing 5. (Schluß)

Programmname:	sc4.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
NmO /*********************************************************************
Sc6       Supercopy v1.0 (w) in 1988 by Eichenberger Software Production
CpG                   loeschen und kopieren von Directorys und makedir
XZU                                 TURBO-script creater
N70 Wed Oct  5 14:34:15 1988
Az  ****************************************************SC4*/
rG  /*========================
uV4    HEADERS
JB1  ========================*/
Wz0 #include<functions.h>
d0  #include<vd0:sc_struct.h>
wL  /*========================
2w4    DEFINES
OG1  ========================*/
Kx0 #define SOURCE 0L
w6  #define DESTINATION 1L
1Q  /*========================
YA4    GLOBALE VARIABLEN
TL1  ========================*/
OUO extern struct Source src;
8o  extern struct Destination dst;
F1  extern struct Volumes vlm;
EP  extern struct GfxBase *GfxBase;
qi  extern struct RastPort *mainRP;
T2  extern struct Window *Main_Window;
T1  extern struct DosLibrary *DOSBase;
wV  extern struct NewWindow rename_nw;
r4A          struct Window *Rename_Window;
aB0 extern struct NewWindow format_nw;
XzA          struct Window *Format_Window;
IQ0 extern UBYTE  MessageLine[80];
yc  extern UBYTE  r_m_buffer[80];
Uv  extern UBYTE  diskname_buffer[80];
XE  extern struct Gadget MAIN_message;
mE  extern struct Gadget rename_r_m;
ER  extern struct IntuiText format_IText2;
qy  static char *block_pfad[1];
yN  /*********************************************************************
d5X                                  Funktion fuer:
QqN                         Directory loeschen   -> flag = 1
mD                          Directory kopieren   -> flag = 0
4p0 Wed Oct  5 14:34:07 1988
8X  *********************************************************************/
BZ  CopyDir(directory,flag,move)
EE2   char directory[];
NB3    ULONG flag;
kn     BOOL move;
rW     struct FileLock *dirlock,*oldlock;
U5     WORD success,check;
Ld     register int i,t;
cC     if((block_pfad[0][0] == NULL)){
L16       if(!(block_pfad[0]=AllocMem(300L,MEMF_CHIP|MEMF_CLEAR)) ||
rS9          !(block_pfad[1]=AllocMem(300L,MEMF_CHIP|MEMF_CLEAR))){
6N           WriteMessage("no memory free !");
fs3    namecat("",block_pfad[SOURCE],SOURCE);
en     namecat(directory,block_pfad[SOURCE],3L);
9n     namecat("",block_pfad[DESTINATION],DESTINATION);
13     namecat(directory,block_pfad[DESTINATION],3L);
tH     f(flag == 1){
4r6       if((dirlock=Lock(block_pfad[checkblocks()],ACCESS_READ))==0){
Jc9          WriteMessage("can't lock selected dir !");
TQ6       goto end;
eZ3    else{
996       if((dirlock=Lock(block_pfad[SOURCE],ACCESS_READ))==0)
          {
ng9          WriteMessage("can't lock selected dir !");
XU6       goto end;
YO3    oldlock=CurrentDir(dirlock);
JN     GetDir(dirlock,flag,move);
DM     end:;
5e     oldlock=CurrentDir(oldlock);
JT     UnLock(dirlock);
qB     if(flag == 1){
kQ6       if(DeleteFile(block_pfad[checkblocks()])==0){
D09          detect_errors();
Gr           return(-1);
Xh3    block_pfad[0][0]=0;
iE     FreeMem(block_pfad[0],300L);
pH     FreeMem(block_pfad[1],300L);
zD0 GetDir(curr_lock,flag,move)
x52   struct FileLock *curr_lock;
wk3    ULONG flag;
JM     BOOL move;
7T     struct FileLock *newlock,*oldlock,*ignorelock,*dirlock;
Le     struct FileInfoBlock *fib;
8R     static BYTE loop;
5g     WORD success,check;
wE     register int i,t;
1L     check=checkblocks();
8j     if(flag==1) goto next;
Cb0 /*========================
532   Directory im Destination-
4k3    Dir schon vorhanden ?
fX1  ========================*/
3z3    if( (ignorelock = Lock(block_pfad[DESTINATION],ACCESS_READ)) == NULL){
```

Listing 6. »sc4.c« bitte mit dem Checksummer eingeben

```
616       WriteMessage("creating a new directory.");
Tu        if(!(dirlock=CreateDir(block_pfad[DESTINATION]))){
aN9          detect_errors();
Au6       UnLock(dirlock);
m23    if(ignorelock != NULL) UnLock(ignorelock);
RuO next:
sM3    fib = (struct FileInfoBlock *)
1P9          AllocMem((long)sizeof(struct FileInfoBlock),MEMF_C
             HIP|MEMF_CLEAR|MEMF_PUBLIC);
yW3    success=Examine(curr_lock,fib);
ez     i=0;
zy     if(flag==0) check=1;
ko2   while (success != 0)
xZ6       success = ExNext(curr_lock,fib);     /* examine the n
          ext entry */
5h        if(success != NULL){
Jq9          if((fib->fib_EntryType > 0) && (success != NULL)
             ){
XaC             namecat(&fib->fib_FileName[0],block_pfad[check
                ],3L);
0g              newlock=Lock(&fib->fib_FileName[0]);/* neues D
                irectory 'locken' */
Fj              oldlock=CurrentDir(newlock);        /* Director
                y aktuell machen */
Un              GetDir(newlock,flag,(BOOL)0);
Ek              UnLock(newlock);
Xs              if(flag == 1){
0QF                if(DeleteFile(block_pfad[check])==0){
uhI                   detect_errors();
xY                    return(-1);
1AC             oldlock=CurrentDir(curr_lock);
bL              if(flag==1) Examine(curr_lock,fib); /* nur wenn
                 im Delete-Modi */
cw              Parent_Dir(block_pfad[check],2L);
bW9          else{
EbC             if(flag==0){
Z5F                if(copy_file(block_pfad[check],&fib->fib_Fi
                   leName[0]) == -1){
URI                   goto end;
i3C             if(flag == 1){
p9F                if(fib->fib_EntryType <= 0){
mFI                   if(delete_files(block_pfad[check],&fib->
                      fib_FileName[0],fib,curr_lock)==-1){
YVC             goto end;
Dm              if(move == 1){
tDF                if(fib->fib_EntryType <= 0){
yhI                   if(delete_files("",&fib->fib_FileName[0]
                      ,fib,curr_lock)==-1){
cZ                    goto end;
wu9          i++;
fYO end:
Iz3    FreeMem(fib,(long)sizeof(struct FileInfoBlock));
c80 copy_file(dst,source)
6n3    char dst[],source[];
D4     register struct FileHandle *fh1,*fh2;
Gs     char filename[500],destination[500];
ig     struct FileLock *oldlock;
2p     char *file_buffer;
mn     LONG error;
Qm     LONG count,count1;
pv     struct FileHandle *Open();
ZQ     sprintf(destination,"%s/%s",dst,source);
ED     WriteMessage(source);
d1     if((fh1 = Open(source,MODE_OLDFILE))==0){   /* source nam
       e */
1u6       puts("can't open");
PP        error = IoErr();
mM        goto cleanup;
6b3    if((fh2 = Open(destination,MODE_NEWFILE))==0){   /* dst n
       ame */
py6       puts("can't open");
TT        error = IoErr();
qQ        goto cleanup;
5i3    file_buffer = (char *)AllocMem(11264L,MEMF_CLEAR);
9g     while(count=Read(fh1,file_buffer,5632L)){
Yv6       if(count == -1){
Jo9          WriteMessage("Read-Error - corrupt File !");
Re           FreeMem(file_buffer,11264L);
Xm           Close(fh2);Close(fh1);
fG           return(-1);
EP6       if(Write(fh2,file_buffer,(LONG)count) != count){
jB9          WriteMessage("Write-Error");
Wj           FreeMem(file_buffer,11264L);
cr           Close(fh2);Close(fh1);
kL           return(-1);
5b0 cleanup:
kt3    if(file_buffer) FreeMem(file_buffer,11264L);
Ca     if(fh2) Close(fh2);
8U     if(fh1) Close(fh1);
v00 } /* end of block */
1s  delete_files(dst,source,fib,lock)
gN3    char dst[],source[];
o7     struct FileInfoBlock *fib;
hK     struct FileLock *lock;
rT     char filename[500],destination[500];
GP     register ULONG success;
i42   if(dst[0] != 0){
S16      switch(checkblocks()){
Mw9         case SOURCE:
whF               if(source[0] != NULL)
68                if(DeleteFile(source)==0){
zmI                  detect_errors();
2d                   return(-1);
03F               success=Examine(lock,fib);
w5                break;
hN9         case DESTINATION:
3oF               if(source[0] != NULL)
H8I                  sprintf(destination,"%s/%s",dst,source);
uhF               else
QzI                  strcpy(destination,dst);
fqF               if(DeleteFile(destination)==0){
9wI                  detect_errors();
Cn                   return(-1);
ADF               success=Examine(lock,fib);
6F                break;
fa6      return(0);
q13    else{
NP6       if(DeleteFile(source)==0){
G39          detect_errors();
Ju           return(-1);
HK6       success=Examine(lock,fib);
1g        return(0);
SXO } /* end of block */
E6  write_turbo_script(){
Q13    FILE *fp;
Be     char pfad[500];
Ls     register int i;
sc     fp=fopen("script.asc","a");
xG     switch(checkblocks()){
rR6       case SOURCE:
V1        for(i=0;i<src.list.entrys;i++){
3zC             if((src.list.select[i] == 1) && (src.list.dir[i
                ] != 1)){
2dF                namecat(src.list.name[i],pfad,SOURCE);
Pd                 fprintf(fp,"%s\n",pfad);
Aq6       case DESTINATION:
FY        for(i=0;i<dst.list.entrys;i++){
y8C             if((dst.list.select[i] == 1) && (dst.list.dir[i
                ] != 1)){
mRF                namecat(dst.list.name[i],pfad,DESTINATION);
Ui                 fprintf(fp,"%s\n",pfad);
dd3    }/*end of switch*/
jb     fclose(fp);
(C) 1988 M&T
```

Listing 6. (Schluß)

Programmname:	sc5.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
NmO /*******************************************************
    ******************
Sc6       Supercopy v1.0 (w) in 1988 by Eichenberger Software P
          roduction
KdR                            Formatieren einer Diskette
EnO Sat May 14 17:15:41 1988
3w  *******************************************************
    *************SC3*/
qF  /*========================
tU4    HEADERS
IA1  ========================*/
Vy0 #include<functions.h>
42  #include<devices/trackdisk.h>
d0  #include<vd0:sc_struct.h>
wL  /*========================
2w4    DEFINES
OG1  ========================*/
```

```
Kx0 #define SOURCE 0L
w6  #define DESTINATION 1L
1Q  /*========================
YA4     GLOBALE VARIABLEN
TL1  ========================*/
0U0 extern struct Source src;
8o  extern struct Destination dst;
F1  extern struct Volumes vlm;
EP  extern struct GfxBase *GfxBase;
qi  extern struct RastPort *mainRP;
Sk  extern struct DosLibrary *DOSBase;
DL  extern UBYTE  MessageLine[80];
tX  extern UBYTE  r_m_buffer[80];
Pq  extern UBYTE  diskname_buffer[80];
S9  extern struct Gadget MAIN_message;
h9  extern struct Gadget rename_r_m;
c2  extern struct Image FRM_Image2;
JH  extern struct Image FRM_Image9;
mt  extern struct Image FRM_drive;
Qq  extern struct Image FRM_format_rot;
Ys  extern struct Image FRM_format_weiss;
Me  extern struct Image verify_img;
Ce  extern struct Border FRM_Border4;
cK  extern struct NewWindow FRM_nw;
8h  extern struct NewWindow rename_nw;
iH  extern struct Window *Main_Window;
iAA          struct Window *Format_Window;
5I           struct Window *Rename_Window;
sr0 extern VOID drawbox();
So  extern struct Gadget FRM_diskname;
wd  #define FRM_ImageList1 FRM_Image2
BV  #define FRM_BorderList1 FRM_Border4
WK  #define       NUMCYLS 80            /*  normal # of cylinders */
1Y  #define       MAXCYLS (NUMCYLS+20)  /* max # cyls to look for during cal */
Nq  #define       NUMHEADS 2
4B  #define       NUMTRACKS (NUMCYLS*NUMHEADS)
JW  #define BLOCKSIZE TD_SECTOR
hn  #define CYLSIZE BLOCKSIZE*NUMHEADS*NUMSECS
py  #define MOTOR_ON 1
cE  #define MOTOR_OFF 0
EI  struct MsgPort *diskport;
lk  struct IOStdReq *diskreq;
n2  ULONG openerror = -999;
QL  ULONG Block;
hF  char mask[]={1,4,9};/**/
uR  unsigned char *disk_memory,*abfall_memory;
ae  ULONG *buf;
M1  /***********************************************************************
                           Format Requester init. und auf Aktionen warten
Ts0 ***********************************************************************/
nV  InitFormatReq(){
KW3    register LONG success;
o82   struct IntuiMessage *Message;
U9    register WORD mouse_x,mouse_y;
Y23    register struct RastPort *rp,*Open_FormatWindow();
3I     struct IntuiMessage *Wait_for_Message();
zB     BOOL verify_flag=1;
aT2   LONG drive=0,test_drive,Check_connected_Drives();
9V3    LONG gadid;
JL     ULONG count;
JD     rp=Open_FormatWindow();
k02   test_drive=Check_connected_Drives();
Fi3    ActivateGadget(&FRM_diskname,Format_Window,NULL);
gq     FOREVER{
1r6       Message=Wait_for_Message(Format_Window,1L);
DY5      gadid = (USHORT)((struct Gadget *) Message->IAddress)->GadgetID;
6j6       if(gadid == CANCEL){
Nf        CloseWindow(Format_Window);
0Z9          return(FALSE);
ct6       if(Message->Class == RAWKEY){
H09          switch(Message->Code){
hwC             case 0x34:  /* fuer 'V' */
cdF                if(verify_flag == 1){
FeI                   verify_flag=0;
Xb                    DrawImage(rp,&verify_img,0L,0L);
50E               else{
MmI                   verify_flag=1;
Gs                    RectFill(rp,232L,52L,(long)232+30,(long)52+14);
QZF                break;
A16       if((gadid == OK) || (gadid == DISKNAME)){
s59          FormatingDisk(drive,diskname_buffer,verify_flag,rp);
bt6       CloseWindow(Format_Window);
KQ9          /*FormatingDisk(drive,diskname_buffer,verify_flag,rp);*/
Jp           goto quit;
J86       if(Message->Class == MOUSEBUTTONS){
i59          mouse_x=Message->MouseX;
qF           mouse_y=Message->MouseY;
9Y           if((mouse_x >= 75) && (mouse_x <= 158) &&
EFC             (mouse_y >= 36) && (mouse_y <= 45)){
YUA           count = (mouse_x-75)/28; /* 0, 1, 2 */
DIC             if((test_drive&mask[count])==0) goto weiter; /**/
daA           SetAPen(rp,0L);
TMC             RectFill(rp,80L,25L,170L,35L);  /* aktelles drive loeschen*/
ry              DrawImage(rp,&FRM_drive,(long)(count*30),0L);
wQ              drive=count;
zK6       weiter:
W30 quit:
AZ  /***********************************************************************
                                  Formatieren einer Diskette
Hg0 ***********************************************************************/
3J  FormatingDisk(Drive,diskname,verify_flag,rp)
0d3   ULONG Drive;
pe    UBYTE diskname[];
Np2  BOOL verify_flag;
DT3   struct RastPort *rp;
qW    register ULONG i,n;
vh2   char text[3];
ht3   LONG Wait_on_Message();
11    Open_All(Drive);
Rp    SwitchMotor(MOTOR_ON);
gn    for(i=0;i<NUMCYLS;i++){
Oy5     SetAPen(rp,1L);
Qf6     SetDrMd(rp,JAM2);
Ew      sprintf(text,"%ld",i);
An      Move(rp,200L,63L);
aT      Text(rp,text,(long)strlen(text));
KL      if(verify_flag == 1){
kG9        DrawImage(rp,&FRM_format_rot,0L,0L);
G0         FormatCylinder(i,disk_memory);
7y         if(Wait_on_Message(Format_Window) == CANCEL){
ebC           goto end;
mk9        DrawImage(rp,&FRM_format_weiss,0L,0L);
Lz         DrawImage(rp,&FRM_format_rot,0L,8L);
GS         if(Read_DiskBlock(i,abfall_memory,(BOOL)1)==-1){
aGC           puts("Verify_Error !!!");
c49        DrawImage(rp,&FRM_format_weiss,0L,8L);
up6     else{
uQ9        DrawImage(rp,&FRM_format_rot,0L,0L);
QA         FormatCylinder(i,disk_memory);
H8         if(Wait_on_Message(Format_Window) == CANCEL){
o1C           goto end;
wu9        DrawImage(rp,&FRM_format_weiss,0L,0L);
Io3    W.ite_DiskInformation(diskname);
s10 end:
s03    SwitchMotor(MOTOR_OFF);
mK     cleanup(NULL);
Ba0 /*========================
2Y4     alles freigeben
dV1  ========================*/
hC0 cleanup(mes)
1U  char *mes;
BL3    if (mes) puts(mes);
wV     /* drive df0: wieder freigeben */
Cx     if(!openerror) CloseDevice(diskreq);
cr     if(diskreq) DeleteStdIO(diskreq);
F6     if(diskport) DeletePort(diskport);
2D     if(disk_memory) FreeMem(disk_memory, (ULONG)CYLSIZE);
7o     if(abfall_memory) FreeMem(abfall_memory, (ULONG)CYLSIZE);
Nm0 /*========================
So4     Drive ein/aus
ph1  ========================*/
z10 SwitchMotor(onoffswitch)
LU  ULONG onoffswitch;
tI3    diskreq->io_Length = onoffswitch;
```

Listing 7. »sc5.c« bitte mit dem Checksummer eingeben

```
169 xo      diskreq->io_Command = TD_MOTOR;
170 uC      DoIO(diskreq);
171 w70 /* lesen eines Blocks von Diskette */
172 Eh  Read_DiskBlock(block, buf,flag)
173 pG  ULONG block;
174 pQ  unsigned char *buf;
175 Fp  BOOL flag;
176 1k3     diskreq->io_Length = BLOCKSIZE;
177 VV      if(flag == 1) diskreq->io_Length = CYLSIZE;
178 1e      diskreq->io_Data = (APTR) buf;
179 Pe      diskreq->io_Command = CMD_READ;
180 nC      diskreq->io_Offset = block*BLOCKSIZE;
181 ct      if(flag == 1) diskreq->io_Offset = block*CYLSIZE;
182 60      DoIO(diskreq);
183 Jg      if(diskreq->io_Error != 0){
184 nt6        puts(" *** Read-Error ***");
185 zl         printf("-- Block nr. %d   --- Error Number is: %d\n",
           block,diskreq->io_Error);
186 3e         return(-1);
187 UP3     return(0);
188 mB0 /*========================
189 Q44     schreiben eines Blocks
190 E61  ========================*/
191 L20 Write_DiskBlock(block,buf)
192 Is2   int block;
193 AS3     UBYTE *buf;
194 J2      diskreq->io_Length = BLOCKSIZE;
195 Iv      diskreq->io_Data = (APTR) buf;
196 ao      diskreq->io_Command = CMD_WRITE;
197 4T      diskreq->io_Offset = block*BLOCKSIZE;
198 Me      DoIO(diskreq);
199 zw      if(diskreq->io_Error != 0){
200 Ws6        puts(" *** Write-Error ***");
201 F1         printf("-- Block nr. %d   --- Error Number is: %d\n",
           block,diskreq->io_Error);
202 OPO /*========================
203 Q14     Updaten eines Blocks
204 SK1  ========================*/
205 mWO UpDate()
206 z83     diskreq->io_Command = CMD_UPDATE;
207 Vn      DoIO(diskreq);
208 85      if(diskreq->io_Error != 0){
209 AG6        puts(" *** UpDate-Error ***");
210 hZ         printf("Error Number is: %ld\n",diskreq->io_Error);
211 9YO /*========================
212 XC1  Track formatieren -> ohne Verify
213 bT   ========================*/
214 690 FormatCylinder(cyl, buf)
215 pq3     LONG cyl;
216 Xp      UBYTE *buf;
217 wH      diskreq->io_Length = CYLSIZE;
218 fI      diskreq->io_Data = (APTR) buf;
219 iK      diskreq->io_Command = TD_FORMAT;
220 fd      diskreq->io_Offset = cyl*CYLSIZE;
221 j1      DoIO(diskreq);
222 MJ      if(diskreq->io_Error != 0){
223 GO6        puts(" *** FORMAT-Error ***");
224 vn         printf("Error Number is: %ld\n",diskreq->io_Error);
225 NmO /*========================
226 8Z  Track formatieren -> mit Verify
227 ph1  ========================*/
228 zh0 Verify_Format(cyl, buf)
229 343     LONG cyl;
230 13      UBYTE *buf;
231 AV      diskreq->io_Length = CYLSIZE;
232 tW      diskreq->io_Data = (APTR) buf;
233 j9      diskreq->io_Command = ETD_FORMAT;
234 tr      diskreq->io_Offset = cyl*CYLSIZE;
235 xF      DoIO(diskreq);
236 aX      if(diskreq->io_Error != 0){
237 UE6        puts(" *** FORMAT-Error ***");
238 91         printf("Error Number is: %ld\n",diskreq->io_Error);
239 b00 /*========================
240 aD2   Pruefsumme neu berechnen
241 3v1  ========================*/
242 4v0 CorrectBlockChecksum(buf)
243 yG  UBYTE *buf;
244 uS3     ULONG checksum, *long_ptr;
245 Qs      LONG i;
246 jH      long_ptr = (ULONG *)buf;
247 pC      checksum = 0;
248 8n      for(i=0; i< BLOCKSIZE/4;i++)
249 7x6        checksum += long_ptr[i];
250 jm3     long_ptr[5] -= checksum;
251 nC0 /*========================
252 Qx4     Block besetzen
253 F71  ========================*/
254 190 Alloc_DiskBlock(block, buf)
255 Jt  int block;
256 BT  UBYTE *buf;
257 s23     ULONG *long_ptr = (ULONG *) buf;
258 0Q      LONG longword, bit, i;
259 DD      ULONG checksum;
260 TO      longword = (block-2)/32;
261 x5      bit = block-2-longword*32;
262 SU      long_ptr[longword+1] &= **ß(1L<<bit);
263 b9      /* BAM checksumme berechnen */
264 pH      for(i=1,checksum = 0;i<BLOCKSIZE/4;i++)
265 ND6        checksum += long_ptr[i];
266 Sm3     long_ptr[0] = -checksum;
267 3SO /*========================
268 Jo4     oeffne Device, Port usw.
269 VN1  ========================*/
270 c70 Open_All(drive)
271 XL3     LONG drive;
272 xS      /* CHIP Memory fuer buf holen */
273 zk5      if(!(disk_memory = AllocMem((LONG)CYLSIZE,MEMF_CHIP|ME
         MF_CLEAR)))
274 NuD            cleanup("No memory !");
275 2U5      if(!(abfall_memory = AllocMem( (LONG)CYLSIZE,MEMF_CHIP
         |MEMF_CLEAR ) ) )
276 PwD            cleanup("No memory !");
277 IZ3     /* Port zur Kommunikation mit trackdosk.device aufbauen
        */
278 Gt5      if(!(diskport = CreatePort("mydisk.port", 0L)))
279 xk8         cleanup("CreatePort failed!");
280 cH3     /* StdIO-Block zur Kommunikation ueber 'diskport' aufbau
        en */
281 KyA          if(!(diskreq = CreateStdIO(diskport)))
282 C38         cleanup("CreateStdIO failed!");
283 vv3     /* reservieren der trackdisk.device df0: */
284 TR5      openerror = OpenDevice(TD_NAME,(LONG)drive,diskreq,dri
         ve);
285 H98         if(openerror) cleanup("OpenDevice failed!");
286 M10 /*========================
287 k24     mache Disk Doswuerdig!!
288 og1  ========================*/
289 HMO Write_DiskInformation(diskname)
290 cs3     char diskname[];
291 bH      register ULONG i,n;
292 nQ      ULONG *l_buf;
293 Fo      ULONG namelength;
294 m0      for(i=0;i<4;i++)
295 SG6        disk_memory[i] = "DOS\0"[i];
296 m53     Write_DiskBlock(0,disk_memory);
297 7b      l_buf = (ULONG *)disk_memory;
298 ER      l_buf[0] = 2;
299 Er      l_buf[3] = 72;
300 Wh      l_buf[78] = 0xffffffff;
301 jB      l_buf[79] = 881;
302 5L      DateStamp(&l_buf[121]);
303 KI      DateStamp(&l_buf[105]);
304 Ev      disk_memory[432] = strlen(diskname);
305 vN      for(i=0;i<strlen(diskname);i++)
306 B06        disk_memory[433+i] = diskname[i];
307 NU3     l_buf[127] = 1;
308 fA      CorrectBlockChecksum(disk_memory);
309 eF      Write_DiskBlock(880,disk_memory);
310 P1      for(i=1;i<56;i++)
311 br6        l_buf[i] = 0xffffffff;
312 IX3     Alloc_DiskBlock(880,disk_memory);
313 Oe      Alloc_DiskBlock(881,disk_memory);
314 oQ      Write_DiskBlock(881,disk_memory);
315 ju      UpDate();
316 SrO /****************************************************
    *****************
317 u7N                   Format Window oeffnen und zeichnen
318 AoO Sun May 15 10:24:39 1988
319 az  ****************************************************
    *****************/
320 WK  struct RastPort *Open_FormatWindow(){
321 EU2   register struct RastPort *rp;
322 X33     register WORD i;
323 6q      static LONG y1[] = {8,10,12,14,16};
324 dF      static LONG y2[] = {78,80,82,84,86};
325 Ou2   Format_Window=OpenWindow((struct NewWindow *) &FRM_nw);
```

```
326 vu3     rp = Format_Window->RPort;
327 FD      SetAPen(rp,1L);
328 1T      for(i=0;i<5;i++){
329 FK6        Move(rp,9L,(long)y1[i]);
330 tH         Draw(rp,355L,(long)y1[i]);
331 rH         Move(rp,63L,(long)y2[i]);
332 N3         Draw(rp,302L,(long)y2[i]);
333 Aj3     Move(rp,1L,22L);
334 p1      Draw(rp,365L,22L);
335 cI      Move(rp,1L,47L);
336 VP      Draw(rp,365L,47L);
337 VV      SetAPen(rp,3L);
338 ta      for(i=0;i<4;i++){
339 jn6        drawbox(Format_Window->RPort,(long)187+i,(long)51+i,
            (long)187+38-i,(long)51+18-i);
340 cQ3     DrawBorder(rp,&FRM_BorderList1,0L,0L);
341 hI4      DrawImage(rp,&FRM_ImageList1,0L,0L);
342 Ah       DrawImage(rp,&FRM_Image9,0L,0L);
343 x43     DrawImage(rp,&FRM_drive,0L,0L);  /*  */
344 8r      return(rp);
345 YJ0 /* last Update:
346 kT  Wed Oct  5 14:34:41 1988
347 e3  */
348 x9  LONG Check_connected_Drives(){
349 r13     register LONG i,n=0,count=0;
350 C7      static char *drives[] = {"DF0","DF1","DF2"};
351 Gn      for(i=0;i<=29;i++){
352 GE6        for(n=0;n<=2;n++){
353 ym9           if(strcmp(vlm.n[i],drives[n]) == 0){
354 snC              count|=mask[n];/**/
355 5P3     return(count);
(C) 1988 M&T
```

Listing 7. (Schluß)

Programmname:	sc6.c
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	C
Bemerkung:	siehe Listing 1

```
Programmname: sc6.c
-------------------

  1 Nm0 /*********************************************************
        ******************
  2 Sc3    Supercopy v1.0 (w) in 1988 by Eichenberger Software Prod
           uction
  3 KPC               1. Zeigt die freie Speicherkapazitaet eines Lau
                      fwerks an
  4 dT                2. Type und Typeh
  5 7WI                  3. Command
  6 Bt0 Wed Oct  5 14:35:12 1988
  7 Yx  *********************************************************
        *****************/
  8 sH  /*========================
  9 vW4    HEADERS
 10 KC1  ========================*/
 11 X00 #include<functions.h>
 12 e1  #include<vd0:sc_struct.h>
 13 xM  /*========================
 14 3x4    DEFINES
 15 PH1  ========================*/
 16 nU0 #define SPACE                  0x20
 17 qW  #define BACKSPACE              0x08
 18 1Y  #define LINEFEED               0x0a
 19 nX  #define RETURN                 0x0d
 20 We  #define CSI                    0x9b
 21 al  #define DEVICE_STATUS_REPORT   0x36, 0x6e
 22 OS  #define WINDOW_STATUS_REQUEST  0x30, 0x20, 0x71
 23 oX  #define DELETE_LINE            0x4d
 24 Sg  #define CURSOR_UP              0x41
 25 Dn  #define CURSOR_DOWN            0x42
 26 4F  #define CURSOR_Shift_UP        0x54
 27 On  #define CURSOR_Shift_DOWN      0x53
 28 FP  #define ARG1 *(long_memory+counter+1),*(long_memory+counte
        r+2),*(long_memory+counter+3),*(long_memory+counter+4)
 29 YB  #define SOURCE 0L
 30 AK  #define DESTINATION 1L
 31 Lv  UBYTE del_line[] = {CSI,DELETE_LINE,CSI,0x32,CURSOR_UP};
 32 yi  UBYTE change_color[] = {CSI,'2',';','3','3',';','4','0','m'
        ,
 33 CNO                            CSI,'0',';','3','1',';','4','0','m'
                                   };
 34 Ih0 /*========================
 35 pR4    GLOBALE VARIABLEN
 36 kc1  ========================*/
 37 H10 extern struct Source src;
 38 P5  extern struct Destination dst;
 39 WI  extern struct Volumes vlm;
 40 Vg  extern struct GfxBase *GfxBase;
 41 7z  extern struct RastPort *mainRP;
 42 kJ  extern struct Window *Main_Window;
 43 k2  extern struct DosLibrary *DOSBase;
 44 Dm  extern struct NewWindow rename_nw;
 45 8LA          struct Window *Rename_Window;
 46 rS0 extern struct NewWindow format_nw;
 47 oGA          struct Window *Format_Window;
 48 pB7       struct Window *Type_Window;
 49 a10 extern UBYTE  MessageLine[80];
 50 Gu  extern UBYTE  r_m_buffer[80];
 51 mD  extern UBYTE  diskname_buffer[80];
 52 pW  extern struct Gadget MAIN_message;
 53 4W  extern struct Gadget rename_r_m;
 54 Wj  extern struct IntuiText format_IText2;
 55 P4  struct IntuiMessage  *message;
 56 e3  /*========================
 57 6C4    Globale Variablen fuer Type-Window
 58 6y1  ========================*/
 59 iV0 struct MsgPort *Con_w_Port;
 60 ZH  struct MsgPort *Con_r_Port;
 61 IL  struct IOStdReq *Con_w_Msg;
 62 42  struct IOStdReq *Con_r_Msg;
 63 wn  ULONG conErr;
 64 L2  struct NewWindow Type_nw = {0,0,640,256,0,1,GADGETUP+MOUSEB
        UTTONS+CLOSEWINDOW+RAWKEY,
 65 ox3   WINDOWCLOSE+WINDOWSIZING+WINDOWDEPTH+ACTIVATE,NULL,NULL,
          NULL,NULL,NULL,640,20,640,257,
 66 1K    WBENCHSCREEN
 67 OK0 };
 68 xv  struct FileHandle *filehandle=0;
 69 Ts  /*********************************************************
        *****************
 70 1v6      schreibt die aktuelle Diskettenkapazitaet in den 'fre
             e-space' block
 71 uEM                  wenn call = 0  => die zahl addieren
 72 uE                   wenn call = 1  => die zahl subtrahie
                         ren
 73 eE                   wenn call = 2  => aktualisieren
 74 d20 *********************************************************
        *****************/
 75 o1  Write_Info(flag,call,zahl)
 76 rf3   ULONG flag;
 77 91    BOOL call;
 78 Cw    ULONG zahl;
 79 p1    static ULONG actual_diskspace[2];
 80 rq    struct InfoData *infodata;
 81 An4    struct FileLock *lock;
 82 ed2   char bytes[12];
 83 uk3   LONG count;
 84 k12   infodata = (struct InfoData *) AllocMem((long)sizeof(stru
        ct InfoData),MEMF_CLEAR);
 85 v03   SetAPen(mainRP,1L);
 86 pj2   SetDrMd(mainRP,JAM2);
 87 Vf3   switch(flag){
 88 rR6      case SOURCE:
 89 1n9         lock =Lock(src.spath,ACCESS_READ);
 90 JR          Info(lock,infodata);
 91 Y7          if(strncmp(src.spath,"RAM:",4) != NULL){
 92 G7          count=(infodata->id_NumBlocks-infodata->id_NumBl
                ocksUsed)
 93 MMG               * (infodata->id_BytesPerBlock);
 94 L89         else
 95 IWC            count = AvailMem(MEMF_CHIP)+AvailMem(MEMF_FAST)
                   ;
 96 n66      switch(call){
```

Listing 8. »sc6.c« bitte mit dem Checksummer eingeben

```
ZpC             case 0:
1NF                count+=zahl;break;
duC             case 1:
7VF                count-=zahl;break;
or9          if(count>99999999) count==0; /**/
xL           sprintf(bytes,"%81d",count);
YH           Move(mainRP,182L,191L);
ye           Text(mainRP,bytes,(long)strlen(bytes));
278        UnLock(lock);
dm6      break;
04       case DESTINATION:
F89          lock =Lock(dst.dpath,ACCESS_READ);
ck           Info(lock,infodata);
5S           if(strncmp(dst.dpath,"RAM:",4) != NULL){
ZQ           count=(infodata->id_NumBlocks-infodata->id_NumBlocksUsed)
ffG                * (infodata->id_BytesPerBlock);
eR9          else
bpC             count = AvailMem(MEMF_CHIP)+AvailMem(MEMF_FAST);
6P6      switch(call){
s8C          case 0:
KgF             count+=zahl;break;
wDC          case 1:
QoF             count-=zahl;break;
LR9          if(count < 0) count = 0;
Ge           sprintf(bytes,"%81d",count);
7H           Move(mainRP,182+364L,191L);
Hx           Text(mainRP,bytes,(long)strlen(bytes));
LQ           UnLock(lock);
w56      break;
Av3    FreeMem(infodata,(long)sizeof(struct InfoData));
PoO /***********************************************************************
aic                                        TYPE
p00 Tue May 17 19:00:55 1988
Xw  ***********************************************************************/
I4  Type(){
012   struct IntuiMessage *Message,*Wait_for_Message();
Y83    register LONG gadid;
04     ULONG i=0,n=0,q;
dH     UBYTE *mybuffer=0,*Open_type_things();
XI     char Read_Buffer[80];
oM2   char filename[180];
V53    UBYTE zeilen_speicher[180];
Qu     LONG laenge,z=0,FileSize();
o1     Wait_for_Click(Read_Buffer,filename,&laenge);
6Z0 next:
263    mybuffer=Open_type_things(filename,&laenge);
u5     while((i < laenge)){
RR6       while((*(mybuffer+i) != '\n') && (n<180)){
LP9          zeilen_speicher[n++]=*(mybuffer+i++);
EC6       if(n>136){
es9          WriteMessage("incorrect. Use HTYPE !");
CO           Close_type_things(&laenge,mybuffer);
S3           return(-1);
816       zeilen_speicher[n++]='\n';
Qd        zeilen_speicher[n++]=NULL;
a0        n=0;
FD        i++;
sn        ConsoleStringWrite(Con_w_Msg,zeilen_speicher);
p4        z++;
He        q=(Type_Window->Height-27)/8;
d2        if(z==q){
jX9          if(Wait_for_Space((UWORD)0) == 0) goto end2;
5h           z=0;
fd4     }/*if*/
hq6       end:;
y30 end2:
wc3    Wait_for_Space((UWORD)1);
mG     WriteMessage("");
Tf     Close_type_things(&laenge,mybuffer);
2R0 /***********************************************************************
dQc                                        HTYPE
V60 Sun May 22 13:53:33 1988
AZ  ***********************************************************************/
zU  HType(){
Uo2   struct IntuiMessage *Message;
B13    register LONG gadid;
ZH     ULONG i=0,n=0,q,buf=0,counter=0;
EQ2   ULONG *long_memory;
FD3    UBYTE *mybuffer=0;
Bw     char Read_Buffer[80];
S02   char filename[180];
9j3    UBYTE zeilen_speicher[180];
4Y     LONG laenge,z=0,FileSize();
SP     Wait_for_Click(Read_Buffer,filename,&laenge);
kD0 next:
gk3    mybuffer=Open_type_things(filename,&laenge);
0s0 /*****/
i7  /*========================
dN  n = Zaehler fuer 'zeilen_speicher' (UBYTE)
PA  buf = Zaehler fuer 'mybuffer' (UBYTE)
B31 ========================*/
3D3   long_memory = (ULONG *)mybuffer;
ep    while((i < laenge)){
n16      sprintf(&zeilen_speicher[n],"%041d: %081x %081x %081x %081x %081x ]",i*4,*(long_memory+counter),ARG1);
Jp       n+=52;
di       counter += 5;
1g       for(q=0;q<20;q++){
xr9         if((*(mybuffer+buf) <= 31) || (*(mybuffer+buf) >= 128)){
GxC            sprintf(&zeilen_speicher[n],".");
kD             buf++;
7A             n++;
pk9         else{
iSC            sprintf(&zeilen_speicher[n],"%c",*(mybuffer+buf));
oH             buf++;
BE             n++;
w16      sprintf(&zeilen_speicher[n],"] ");n++;
9t       sprintf(&zeilen_speicher[n],"\n");n++;
Mm       i+=5;
IV       zeilen_speicher[n++]=NULL;
Ss       n=0;
75       i++;
kf       ConsoleStringWrite(Con_w_Msg,zeilen_speicher);
hw       z++;
9W       q=(Type_Window->Height-27)/8;
Vu       if(z==q){
kD9         if(Wait_for_Space((UWORD)0) == 0) goto weiter;
xZ          z=0;
XV4    }/*if*/
g10 weiter:
nT3   Wait_for_Space((UWORD)1);
d7    WriteMessage("");
KW    Close_type_things(&laenge,mybuffer);
Hg0 /*========================
Oc2   eroeffnet alles wichtige
jb1 ========================*/
xq0 Open_all_Con_things(){
iA3    if(!(Con_w_Port = CreatePort("ESP-con-dev_write",0L))){
yM4     puts("I can't open the console-writeport !!");
Bs6       kill_all_type();
Wz3    if(!(Con_w_Msg  = CreateStdIO(Con_w_Port))){
7K6       puts("I can't open the console-write-messageport !!");
Ev        kill_all_type();
mU3    if(!(Con_r_Port = CreatePort("ESP-con-dev_read",0L))){
r44     puts("I can't open the console-readport !!");
Hy6       kill_all_type();
e13    if(!(Con_r_Msg  = CreateStdIO(Con_r_Port))){
QD6       puts("I can't open the console-read-messageport !!");
K1        kill_all_type();
qy3    if(!(Type_Window = OpenWindow(&Type_nw))){
p86       puts("I can't open my high quality spezial Window!!!!!!!");
N4        kill_all_type();
S23    Con_w_Msg->io_Data   = (APTR)Type_Window;
Zp     Con_w_Msg->io_Length  = sizeof(*Type_Window);
8a2   if((conErr = OpenDevice("console.device",0L,Con_w_Msg,0L))!=NULL){
WS6       puts("I can't open the console.device");
S9      kill_all_type();
1L3    Con_r_Msg->io_Device = Con_w_Msg->io_Device;
4H     Con_r_Msg->io_Unit   = Con_w_Msg->io_Unit;
h60 /*========================
8a4     alles zurueckgeben
911 ========================*/
cL0 kill_all_type(){
c53    if(!conErr) CloseDevice(Con_w_Msg);
Ow     if(Con_w_Port) DeletePort(Con_w_Port);
vJ     if(Con_r_Port) DeletePort(Con_r_Port);
HB     if(Con_w_Msg)  DeleteStdIO(Con_w_Msg);
jT     if(Con_r_Msg)  DeleteStdIO(Con_r_Msg);
8r     if(Type_Window) CloseWindow(Type_Window);
```

```
rG0 /*========================
1U4       auf Con schreiben
JB1 ========================*/
9U0 ConsoleWrite(StdReq,string,length)
k33   struct IOStdReq *StdReq;
P5    UBYTE string[];
nJ    ULONG length;
Nn    StdReq->io_Command = CMD_WRITE;
9j    StdReq->io_Data    = (APTR)string;
mc    StdReq->io_Length  = length;
4L    DoIO(StdReq);
2R0 /*========================
4G4       auf Con String schreiben
UM1 ========================*/
e40 ConsoleStringWrite(StdReq,string)
vE3   struct IOStdReq *StdReq;
aG    UBYTE string[];
Xx    StdReq->io_Command = CMD_WRITE;
Jt    StdReq->io_Data    = (APTR)string;
bv    StdReq->io_Length  = -1L;
EV    DoIO(StdReq);
Cb0 /*========================
c84       Zeichen lesen
eW1 ========================*/
470 QueueRead(StdReq, buffer, length)
50  struct IOStdReq *StdReq;
Q7  char          *buffer;
63  ULONG          length;
HE3   StdReq->io_Command = CMD_READ;
Mx    StdReq->io_Data    = (APTR)buffer;
7x    StdReq->io_Length  = length;
96    SendIO(StdReq);
z00 /***********************************************************************
YBT                              Warte auf Space-Taste
2z0 Sun May 22 10:06:35 1988
7W  ***********************************************************************/
hX  Wait_for_Space(flag)
u93   UWORD flag;
Fk    ULONG MessageClass;
cp    USHORT code;
2s2   struct IntuiMessage *message;
613   ConsoleWrite(Con_w_Msg,change_color,9L);
gX    if(flag ==0)
6Z6      ConsoleStringWrite(Con_w_Msg,"\n*** Please press SPACE to continue.***");
eR2   else
tI6      ConsoleStringWrite(Con_w_Msg,"\n*** END OF FILE.***");
dM3   ConsoleWrite(Con_w_Msg,&change_color[9],9L);
IS2   FOREVER{
xe4     if (message = (struct IntuiMessage *)
Hr8         GetMsg(Type_Window->UserPort)){
599          MessageClass = message->Class;
0d           code = message->Code;
45           ReplyMsg(message);
g3           switch (MessageClass){
p18         case CLOSEWINDOW :
TOC             return(0);
w5              break;
Ac              case RAWKEY:
QMI                     switch(code){
eKL                        case 64:
TOI                     ConsoleWrite(Con_w_Msg,&del_line[0],2L);
hH                      ConsoleWrite(Con_w_Msg,&del_line[2],3L);
6BR                             ConsoleStringWrite(Con_w_Msg,"\n");
bY                              goto end;
4D                              break;
fuE             case 0x45:
eZR                             return(0);
7GI                     break;
8H                      break;
JH4     }/*if*/
zy3   }/*forever*/
MV    end:;
ok    return(1);
e30 /***********************************************************************
kkT                              Filegroesse bestimmen
JU0 Sun May 22 10:23:25 1988
mB  ***********************************************************************/
KJ  LONG FileSize (name)
0t3   char name[];
ST    BPTR lock;
03    struct FileInfoBlock *fileinfoblock;
hI    fileinfoblock = AllocMem ((LONG)sizeof (struct FileInfoBlock),MEMF_CLEAR);
r4    lock = Lock (name, ACCESS_READ);
wR    if (lock == NULL) return (-1);
mB    if (Examine (lock,fileinfoblock) == FALSE) return (-1);
v2    UnLock (lock);
fP    return (fileinfoblock->fib_Size);
sH0 /***********************************************************************
pEP                       Warte bis Name angeklickt wird
TN0 Sun May 22 11:27:29 1988
0P  ***********************************************************************/
TL  Wait_for_Click(Read_Buffer,filename,laenge)
aM3   char *Read_Buffer;
6s    char *filename;
sM    ULONG *laenge;
WH2  struct IntuiMessage *Message,*Wait_for_Message();
G83   BOOL Block_Controll();
ib    register LONG gadid,FileSize(),err;
M1    WriteMessage("Please select a name to typing.");
8I    FOREVER{
ii6     Message=Wait_for_Message(Main_Window,0);
ax      if(Message == -1) return(-1);
Ir5    gadid = ((struct Gadget *) Message->IAddress)->GadgetID;
rQ6     if(Block_Controll(gadid) == 1) continue;
Le      switch(checkblocks()){
Fp      case SOURCE:
SUC            if( (gadid >=3) && (gadid <=15) ){
nvF               strcpy(Read_Buffer,src.list.name[gadid-3+src.scroll_count]);
g8                goto next;
hUC           else
EnF               err=1;
pyC           break;
aG9         case DESTINATION:
DZC           if( (gadid >= 48) && (gadid <=61) ){
BzF              strcpy(Read_Buffer,dst.list.name[gadid-48+dst.scroll_count]);
nF               goto next;
obC           else
LuF              err=1;
w5C           break;
o39         default: err=1;break;
W66      if(err==1) continue;
tM0 next:
TH3   namecat(Read_Buffer,filename,(long)checkblocks());
1i    *laenge = FileSize(filename);
Ts0 /***********************************************************************
KyB          Oeffne alle console.device Sachen. Lese File in den Buffer
ue0 Sun May 22 11:33:09 1988
b0  ***********************************************************************/
vr  UBYTE *Open_type_things(filename,laenge)
GL3   char filename[];
Sw    ULONG *laenge;
Ww    static UBYTE *buffer;
WN    Open_all_Con_things();
oX    buffer = (UBYTE *)AllocMem(*laenge,MEMF_CLEAR|MEMF_CHIP);
Jo    filehandle = Open(filename,MODE_OLDFILE);
4c    Read(filehandle,buffer,*laenge);
Wf    return(buffer);
g50 /***********************************************************************
PDD          Schliesse alle console.device Sachen. Schliesse File.
6r0 Sun May 22 11:35:08 1988
oD  ***********************************************************************/
ej  Close_type_things(laenge,mybuffer)
tw3   LONG *laenge;
GV    UBYTE *mybuffer;
YP    if(filehandle != NULL) Close(filehandle);
0h    kill_all_type();
m6    if(mybuffer != NULL) FreeMem(mybuffer,*laenge);
(C) 1988 M&T
```

Listing 8. (Schluß)

Ein Cocktail für alle Fälle

Hier sind sie wieder, unsere kleinen nützlichen Helfer: Tips und Tricks für alle Fälle. Es ist garantiert für jeden etwas dabei. Schauen Sie doch mal rein!

Auch in diesem Sonderheft gibt's wieder eine geballte Ladung Tips und Tricks für Sie. Wir haben wieder einmal kräftig in unserer Trickkiste gewühlt. Aber was heißt hier in unserer Trickkiste — natürlich in der unserer Leser. Denn wir bekommen laufend die heißesten Kniffe zugesandt und wollen diese natürlich für alle Leser zugänglich machen.

Sicher haben auch Sie schon mit dem einen oder anderen Kunstgriff dem Amiga auf die Sprünge geholfen. Behalten Sie aber Ihre Kniffe nicht für sich, sondern machen Sie diese auch für den Rest der Leserschaft zugänglich, indem Sie sie an uns schicken. Veröffentlichte Tricks werden natürlich honoriert.

Logo goes Basic...

Wer einfache Logo-Grafik in Basic erzeugen möchte, dem hilft unser kleines Unterprogramm. Mit der Angabe von Winkel, Länge und Farbe wird eine entsprechende Linie von Startpunkt (0,0) aus gezogen.

```
SCREEN 1,320,200,5,1
WINDOW 1,,(0,0)-(300,186),0,1

REM Dieser Aufruf zieht eine Linie von (0,0)
REM zur Mitte des Bildschirms

a = 35
b = 170
c = 3
CALL LG (a,b,c)

SUB LG (winkel,lan,farbe)
  STATIC
  pi = 3.141593
  LOCATE 1.1
  ra = (winkel+Wges)
       *pi/180
  x1 = lan*COS(ra)+x
  y1 = lan*SIN(ra)+y
  LINE(x,y)-(x1,y1),farbe
    x = x1
    y = y1
  Wges = winkel+Wges
END SUB
```

Das Listing enthält bereits ein kleines Beispielprogramm. Es erzeugt eine Linie mit der Länge 170 im Winkel von 35 Grad. Der Winkel ist in der Variablen »a«, die Länge in »b« angegeben, »c« enthält die Farbe. *(Albert Treytl/M. Jobst/rs)*

Startup mit Abfrage

Kopieren Sie die CLI-Befehle nur bei Bedarf in die RAM-Disk. Die CLI-Befehle in der RAM-Disk sparen Zeit und schonen Ihre Disketten — falls Sie mit dem CLI arbeiten. Das Kopieren benötigt jedoch Speicherplatz und verlängert die Boot-Zeit. Benötigen Sie das CLI nicht, sondern möchten mit der Workbench arbeiten, so ist die Installation der RAM-Disk in der »Startup-Sequence« unnötig. Für diesen Fall können Sie sich zwei Startdisketten erstellen: eine ausschließlich für die Arbeit mit dem CLI, die zweite für die Workbench. Das kostet eine Diskette. Es gibt jedoch eine pfiffigere Methode: Bauen Sie eine Abfrage in der »Startup-Sequence« ein. Der Trick, der hierzu angewendet wird, ist die Verwendung der Befehle FAILAT und CD:

Sie müssen Ihre Startdiskette (im allgemeinen eine Kopie der Original-Workbench) vorher präparieren.

1. Erzeugen Sie ein neues Dateiverzeichnis mit dem Namen »n«:

```
MAKEDIR n
```

2. In diesem Verzeichnis erzeugen Sie mit dem Editor ein File mit dem Namen »dummy«:

```
ED n/dummy
```

Es spielt keine Rolle, was in diesem File steht. Schreiben Sie einen Text und verlassen den Editor mit <ESC> <x> <RETURN>.

3. Ergänzen Sie diese Zeilen in der »Startup-Sequence«:

```
ECHO "wollen Sie im CLI bleiben j/n ?"
FAILAT 25
CD >nil: ?
IF NOT EXISTS dummy
  SKIP cli
ENDIF
LOADWB
ENDCLI >nil:
LAB cli
CD :
ECHO "wollen Sie RAMCLI aktivieren ?"
FAILAT 25
CD >nil: ?
IF EXISTS dummy
  SKIP nein
ENDIF
; Ein Beispiel um die RAM-Disk mit
; CLI-Befehlen zu aktivieren
MAKEDIR ram:c
COPY c: ram:c
PATH ram:c ADD
LAB nein
CD :
ECHO "Initialisierung
beendet"
```

Wenn Sie nun Ihre Work-Disk neu starten, fragt der Amiga, ob Sie mit dem CLI und mit den CLI-Befehlen in der RAM-Disk arbeiten möchten. Die Erklärung zum Programm:

Das Dummy-Gerät »NIL:« wird gewinnbringend genutzt. Durch Umlenken der Helpfunktion »?« auf dieses Gerät verfügt der Benutzer im CLI über eine elegante Eingabefunktion. Nach Ausgabe des Helptextes auf das nicht existierende Ausgabegerät können Sie einen Text eingeben. Dieser stellt nun das aktuelle Directory dar (CD-Funktion).

Jetzt sucht der Computer in diesem Directory nach dem File »dummy«. Findet er es, war die Eingabe »n«, da nur im Directory »n« ein File »dummy« existiert. Mit Hilfe dieses Tricks läßt sich auch eine Paßwortkontrolle realisieren. Sie brauchen nur ein Directory zu generieren, dessen Name Ihrem Codewort entspricht. In diesem Verzeichnis muß wiederum das File »dummy« stehen. Wird ein falsches Schlüsselwort eingegeben, so findet der Amiga die Datei nicht und beginnt erneut mit der Abfrage.

Dies hilft auf jeden Fall, um die neugierige Schwiegermutter vom Amiga fernzuhalten — falls sie nicht weiß, daß man die »Startup-Sequence« mit <CTRL D> abbrechen kann. *(Martin Horn/rs)*

Batchfiles beschleunigen

Unter einem Batchfile versteht man eine Textdatei, die CLI-Befehle enthält. Man kann in diese Datei eine Reihe solcher Befehle unterbringen, und mit »EXECUTE Name des Batchfiles« werden diese Kommandos automatisch der Reihe nach ausgeführt. Die CLI-Befehle sind Programme wie jedes andere Anwenderprogramm auch — sie müssen also erst von der Diskette geladen werden (sofern sie sich nicht in der RAM-Disk befinden). Die Ladezeiten fallen besonders ärgerlich auf, wenn in einem Batchfile mehrere ECHO-Befehle (funktioniert ähnlich PRINT in Basic) nacheinander auftauchen, und deshalb jedesmal neu geladen werden müssen. Ähnlich verhält es sich mit anderen Kommandos

bei der täglichen Arbeit mit dem CLI (CD...DIR,CD...DIR und so weiter). Neben der Möglichkeit, die Befehle in die schnelle RAM-Disk zu legen, gibt es eine weitere, das RAM einzubeziehen. Man ordnet dem Laufwerk DF0: einen bestimmten RAM-Bereich als Puffer zu, in dem die zuletzt geladenen Befehle gespeichert werden. So brauchen diese Kommandos nicht neu geladen, sondern können aus dem Puffer gelesen werden. Um mit dem Befehl

```
ADDBUFFER Laufwerk Puffergröße
```

wirklichen Zeitgewinn erzielen zu können, sollte man den Puffer etwa 18 bis 20 KByte groß wählen. Beispiel: ADDBUFFERS df0: 18. Der einzige Nachteil dieser Methode soll jedoch nicht verschwiegen werden. Es ist nur durch einen Reset möglich, diesen Speicherbereich wieder anderweitig zu vergeben — man sollte sich also genau überlegen, wieviel Speicher man dafür »opfert«.

(Michael Baas/rs)

Gut gescrollt ist halb...

Die Funktion »ScrollRaster« aus der »graphics.library« ermöglicht dem (C-)Programmierer, einen rechteckigen Bildausschnitt in eine beliebige Richtung zu scrollen. Der Befehl hat die Syntax

```
ScrollRaster(rastport,deltaX,
deltaY,vonX,vonY,bisX,bisY);
```

»deltaX« und »deltaY« geben die Verschiebung in Punkten an. Hier beginnt auch schon das Problem: Normalerweise müßten positive Delta-Werte eine Verschiebung nach unten beziehungsweise rechts, negative Werte eine Verschiebung nach oben beziehungsweise links bewirken.

Daß dem nicht so ist, scheinen nicht einmal die Entwickler selbst bemerkt zu haben. Jedenfalls wird sowohl im »Amiga-Programmier-Handbuch« als auch im »ROM-Kernel-Manual« der Befehl mit oben genannten Scroll-Richtungen angegeben. Als Programmierer ist man spätestens dann verblüfft, wenn sich der Bildschirm, trotz scheinbar korrekter Syntax in die falsche Richtung davon macht. Erst ein Blick in Anhang A des »ROM-Kernel-Manual« offenbart die korrekte (ungewöhnliche) Syntax des Befehls. Zitat:»Die Funktion scrollt die Pixel auf den Punkt (0,0) zu.« Aha. Die korrekten Delta-Werte lauten also:
nach oben beziehungsweise links: positiver Wert;
nach unten beziehungsweise rechts: negativer Wert.

Achtet man bei der Anwendung dieses recht nützlichen Befehls auf diese »verdrehte« Form, bleibt so mancher Ärger erspart.

(Holger Schemel/M. Jobst/rs)

Bildschirmsplitting

Diese kleine Farbdemonstration splittet den Bildschirm in zwei Hälften. Das Programm demonstriert die Abfrage der Position des Rasterstrahls mit Hilfe der Hardware-Register.

```
 execbase = 4
 forbid = -132
 permid = -138
start:
 movem.l d0-d2/a1/a2/a6,-(SP)
 lea $dff004,a1      ; u.a. Y-POS des Rasterstrahls
 lea $dff180,a2      ; Farbregister
 move.l execbase,a6
 jsr forbid(a6)      ; Multitasking aus
zeilenloop:
 move.l #$2b00,d0    ; >>2b<< = Zeile für Split
                     ; >>1b<< = Minimum
Farbloop:
 move.w #$0000,d1    ;erste Farbe
wait:
 move.l (a1),d2      ;
 and.l #$0001ff00,d2; y-Pos. in d2
 cmp.l d0,d2         ; Zeile erreicht ?
 bne wait
 move.w d1,(a2)      ; neue Farbe ins Register
wait1:
 move.l (a1),d2
 and.l #$0001ff00,d2
 cmp.l d0,d2
 beq wait1           ; nächste Zeile abwarten
 cmp.b #$bc,$bfe001
 beq done            ;linke Maustaste gedrückt ?
 add.w #$100,d1      ;neue Farbe
 cmp.w #$1000,d1     ;Farbe wechselt laufend
 blt wait
 add.l #$0100,d0     ;Splitzeile +1
 cmp.l #$13900,d0    ;größer Maximum?
 blt farbloop
 bge zeilenloop
done:
 move.l execbase , a6
 jsr permid(a6)
 movem.l (sp)+,d0-d2/a1/a2/a6
 rts
```

Das Programm ist mit dem Seka-Assembler geschrieben. Nach dem Start wird zunächst jeder weitere Task gesperrt. Danach verändert das Programm ab einer bestimmten Bildschirmzeile die Hintergrundfarbe. Zur besseren Demonstration wird diese Farbe laufend geändert. Nach Durchlaufen der Farbskala erhöht das Programm die Zeile, in der der Bildschirm gesplittet werden soll. Sie können den Programmablauf jederzeit durch einen Druck der linken Maustaste stoppen. Ein Nachteil des Programms ist, daß der Prozessor auf eine bestimmte Rasterposition wartet und nichts anderes erledigen kann. Noch besser wäre es daher, die Farbe des Hintergrunds mit Hilfe des Coppers ab einer bestimmten Position zu verändern.

(Klaus Kuphal/rs)

Animation in Basic

Auch von Basic aus ist es möglich, fließende Animationen zu erstellen. Der Trick besteht darin, mit zwei verschiedenen Screens zu arbeiten. Während der eine Screen das aktuelle Bild der Animation zeigt, wird auf dem anderen »versteckt« bereits das nächste erstellt. Anschließend werden beide Screens vertauscht, und der Vorgang beginnt von neuem. Das folgende kleine Demo-Programm arbeitet mit diesem Prinzip.

```
SCREEN 1,640,200,1,2                  :REM zwei gleiche
SCREEN 2,640,200,1,2                  :REM Screens öff
                                        nen
WINDOW 1,"",(0,0)-(631,186),28,1 :REM zwei gleiche
WINDOW 2,"",(0,0)-(631,186),28,2 :REM Windows öffnen

Variablen:
    x=10:y=10:x1=621:y1=176:w=1:p=1:f=1
Loop:
    WINDOW w                          :REM Window
                                      Nr. w nach vorne
    w=w+1:IF w=3 THEN w=1
    WINDOW OUTPUT w                   :REM Ausgabe
                                 auf Hintergrundwindow
    CLS                               :REM Window
                                      löschen
      LINE (x,y)-(x1,y1),1,b            :REM Rechteck
                                         zeichnen
      x=x+f:y=y+p:x1=x1-f:y1=y1-p   :REM Rechteck
                                         verkleinern
      IF y=170 THEN p=0
      IF x=626 THEN f=-2:p=-1
      IF y=0 THEN p=1
      IF x=0 THEN f=2
    GOTO Loop
```

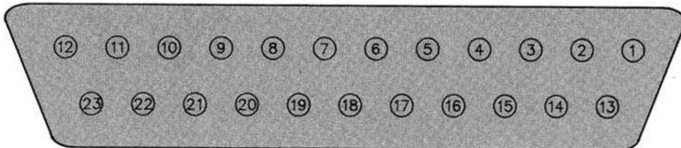
Das Programm erzeugt eine einfache Animation eines sich in der Größe verändernden Rechtecks.

(H. Kalläne/M. Jobst/rs)

Der Schalter für das Laufwerk

Einige Amiga-Programme funktionieren nur, wenn kein externes Laufwerk angeschlossen ist. Aber einige Laufwerke besitzen keinen Ausschalter. Damit Sie sich die Arbeit ersparen, das Laufwerk immer herauszuziehen, bauen Sie sich den Schalter selbst: Sie müssen ihn nur in die RDY-Leitung einlöten — entweder im Laufwerk oder im Anschlußkabel. Das RDY-Signal liegt beim Amiga 500 an Pin 1 des externen Diskanschlusses. Natürlich dürfen Sie den Schalter nur bei ausgeschaltetem Computer betätigen.

(Maurice AL-Khaliedy/rs)



SUB-D-Buchse			23pin/weiblich
Pin	**Signal**	**Beschreibung**	
1	RDY	Bereitschafts-Signal	(active low)
2	DKRD	Schreib/Lese-Signal	(active low)
3	GND	Signalmasse	
4	GND	Signalmasse	
5	GND	Signalmasse	
6	GND	Signalmasse	
7	GND	Signalmasse	
8	MTRXD	Motor-Steuerung	(active low)
9	SEL2B	Laufwerk 2 aktivieren	(gepuffert, active low)
10	DRESB	Reset-Signal	(gepuffert, active low)
11	CHNG	Diskettenwechsel-Signal	(active low)
12	+5	+5 Volt	
13	SIDEB	Kopfwahl-Signal	(gepuffert, active low)
14	WPRO	Schreibschutz-Signal	(active low)
15	TK0	Spur 0-Signal	(active low)
16	DKWEB	Schreibfreigabe-Signal	(gepuffert, active low)
17	DKWDB	Daten schreiben	(gepuffert, active low)
18	STEPB	Signal für Kopf-Stepper	(gepuffert, active low)
19	DIRB	Stepper-Richtungs-Signal	(gepuffert, high = Richtung Spur 0)
20	NC	Nicht belegt	
21	SEL1B	Laufwerk 1 aktivieren	(gepuffert, active low)
22	INDEX	Index-Impuls	(active low)
23	+12V	+12 Volt	

Diese Tabelle zeigt die Belegung des 23poligen SUB-D-Steckers für die Zusatzlaufwerke am Amiga

Super-Hochauflösung

Überall steht geschrieben, der Amiga hätte eine maximale Bildschirmauflösung von 640 x 512 Punkten. In Wahrheit steht Ihnen eine weit höhere Auflösung zur Verfügung. Ein Screen kann bis zu 704 x 564 Punkte groß sein. Die 69376 zusätzlichen Punkte können Sie mit Hilfe der Grafikbefehle des Betriebssystems ohne weiteres nutzen. Besonders in C und Assembler ist dies recht einfach. Zwei Zahlen müssen geändert werden:

Um einen Screen zu öffnen, müssen Sie eine »NewScreen«-Struktur erstellen. Der dritte Wert dieser Tabelle gibt die Screen-Breite an, der vierte die Höhe. Üblicherweise stehen dort, beispielsweise für einen hochauflösenden Screen im Interlace-Modus die Werte 640 und 512. Ersetzen Sie diese Werte durch 704 beziehungsweise 564.

```
NewScreen.Width=704;
NewScreen.Height=564;
```

Wenn Sie nun die Funktion »OpenScreen« anspringen, erscheint der übergroße Screen. Er ist aber nur zum Teil sichtbar. Um dem abzuhelfen, müssen Sie folgendes ändern:

Erstens sollten Sie das Bild des Monitors etwas verkleinern und neu zentrieren. Die Regler hierzu befinden sich im allgemeinen an der Rückseite des Monitors.

Zweitens müssen Sie das Bild mit Hilfe der »Preferences« nach oben links verschieben. Klicken Sie hierzu das große Rechteck in der Mitte des »Preference-Windows« an und bewegen das gesamte Fenster mit der Maus in die gewünschte Ecke. Für die genaue Einstellung ist ein wenig Experimentieren notwendig. Sie dürfen das Bild nicht zu weit verschieben, da sonst der obere Rand des Screens verschwindet. Wenn Sie den Monitor einmal eingestellt haben, können Sie in Ihren Programmen sofort die »Super-Hochauflösung« ausprobieren.

Der Effekt läßt sich im übrigen auch in allen anderen View-Modi ausnutzen, um mehr Punkte darzustellen. So stehen Ihnen zum Beispiel für einen LowRes-Screen 352 statt der üblichen 320 Punkte in der Horizontalen zur Verfügung.

Warum nicht gleich die volle Auflösung?

Die Einschränkung auf 640 x 512 Punkte erfolgt, weil nicht alle Monitore diese extreme Auflösung darstellen können (Overscanning). Besonders an den Rändern kommt es zu Unregelmäßigkeiten. Außerdem sind in den Grenzbereichen nicht alle Sprites sichtbar.

(Jürgen Brendel/rs)

Beckertext und Bilder

Die Bildverarbeitung mit Beckertext ist bei der Verwendung des Hilfsprogramms »BTSnap« aufgrund des hohen Speicherbedarfs der Bilder stark eingeschränkt. Vor allem dann, wenn man nur 512 KByte Hauptspeicher zur Verfügung hat. Erstellt man jedoch seine Grafiken mit DPaint II und speichert die gewünschten Bildausschnitte als Pinsel (Brush), kann eine Menge Speicherplatz eingespart werden. Hinter dem Namen des Pinsels sollte man das Suffix ».IFF« setzen, damit das Bild von Beckertext erkannt wird.

(Jens Scheifgen/M. Jobst/rs)

Tolles Tool mit Anleitung

Der Editor »MicroEmacs« schlägt den ED um Längen. Allerdings nur wenige wissen, daß sich dieses hervorragende Werkzeug auf der ExtrasD-Diskette von Commodore befindet. Schauen Sie nach. Das Programm hat seinen Platz in der Schublade »Tools«. Und auch eine Anleitung ist auf der Diskette gespeichert. Sie verbirgt sich hinter dem Piktogramm »MicroEmacs.doc«. Es existieren drei Wege, die Anleitung zu lesen:

— Klicken Sie das Piktogramm »MicroEMACS.doc« zweimal an. Mit einem System-Requester fordert der Amiga Sie nun auf, das Original der ExtrasD-Diskette einzulegen. Sofern Sie nur ein Laufwerk besitzen, gelangen Sie nach zwei weiteren Diskettenwechseln in das erste Bild der Anleitung.

— Wer die Anleitung schneller lesen möchte, lädt das Utility »More«. Es dient dazu, ASCII-Dateien auf dem Bildschirm auszugeben. Nach der Frage »Filename?«, tippen Sie »MicroEMACS.doc« und drücken <RETURN>.

— Der elegante Weg ist, eine <Shift>-Taste zu drücken und mit der Maus die Piktogramme MicroEMACS.doc und More anzuklicken. Danach wählen Sie in der Menüzeile der Workbench »Open«. Wer es noch einfacher möchte, klickt bei gehaltener <Shift-Taste> MicroEMACS.doc an und aktiviert dann MORE mit einem Doppelklick.

Nach dem Laden sehen Sie auf dem Bildschirm die erste Seite der Anleitung. Durch Drücken von <h> gelangen Sie in einen Screen mit Hilfsanweisungen zu More. Mit <Space> blättern Sie in der Anleitung weiter. Sie umfaßt 70 Seiten und ist in englisch geschrieben. Am Ende befindet sich eine alphabetische Liste aller »Shortcuts« (abgekürzte Befehle) von MicroEMACS. Sie verlassen die Anleitung mit <q>. Wer den gesamten Text intensiv studieren möchte, kann ihn auch mit PrintFiles ausdrucken lassen.

(Nick Seggelke/rs)

Basic und Textverarbeitung?

Haben Sie sich nicht auch schon über den langsamen Editor des Amiga-Basic geärgert? Gerade wenn mit dem List-Fenster gearbeitet wird, dauert es in der Regel unzumutbar lange, bis der Text sich wieder der Größe des Fensters angepaßt hat.

Besser ist es daher, die Programme mit einem herkömmlichen Editor oder einer Textverarbeitung zu schreiben. Durch die intensive Nutzung der Funktionstastenbelegung oder Such- und Ersetzfunktionen der Textverarbeitungsprogramme läßt sich viel

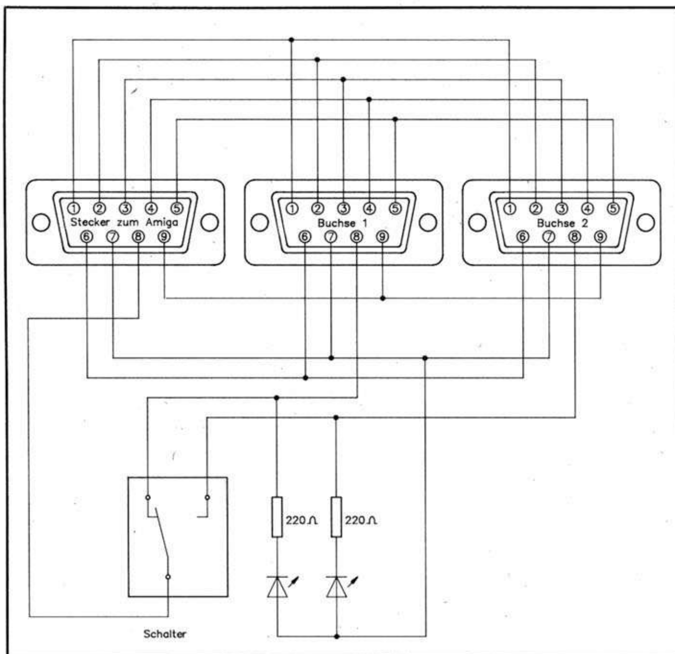
Zeit einsparen. So können Sie beim Schreiben des Programms anstatt des Schlüsselwortes GOSUB die Folge »gb« einsetzen und diese vor dem Speichern durch GOSUB ersetzen. Ein Nachteil dieser Methode ist allerdings das Speichern und erneutes Laden in den Basic-Speicher vor der Ausführung des Programms. Folgendes ist jedoch zu beachten: Sowohl vom Basic-Interpreter als auch von den Textverarbeitungsprogrammen müssen die Programme im ASCII-Format gespeichert werden. Im Basic läßt sich dies durch die Angabe von »,a« im Anschluß an den Dateinamen des SAVE-Befehls erreichen. *(Lutz Beyert/rs)*

Maus rein — Maus raus

Das kann selbst ruhige Zeitgenossen manchmal zur Verzweiflung bringen — immer wenn die Maus angeschlossen ist, braucht man einen Joystick und kurz darauf will ein anderes Programm nur mit der Maus arbeiten. Das damit verbundene häufige Umstecken der Stecker ist nicht nur lästig, es schadet auf lange Sicht auch den Kontakten am Joystick-Port des Amiga. Da liegt es nahe, sich einen Umschalter zu bauen. Hierzu benötigen Sie:

— einen 9-Pin-D-SUB-Stecker
— zwei 9-Pin-D-SUB-Buchsen
— zwei LEDs (rot 18 mA)
— zwei Widerstände 220 Ω
— einen Wechselschalter

Die Bauteile verbinden Sie nach dem im Bild gezeigten Schaltplan. Den fertigen Umschalter schließen Sie mit dem Stecker an den Port des Amiga an. In die Buchsen 1 und 2 stecken Sie den Joystick beziehungsweise die Maus ein. Über den Schalter können Sie nun einfach zwischen beiden hin- und herschalten. Die Leuchtdiode zeigt Ihnen jeweils an, wer aktiv ist. *(Maurice Al-Khaliedy/rs)*



Der Schaltplan für den Maus/Joystick-Umschalter

Datum mit Format

Die Basic-Funktion »DATE$« gibt das aktuelle Datum an. Leider erfolgt die Ausgabe nur im englischen Format. Anstatt »14.12.1988« heißt es »12-14-1988«. Mit folgendem Unterprogramm wird das anders:

```
Datum:
   IF LEN(DATE$)=10 THEN
     Dat$=MID$(DATE$,4,2)+"."+LEFT$(DATE$,2)+"."+RIG
HTS(DATE$,4)
   END IF
   IF LEN(DATE$)=9 AND LEFT$(DATE$,2)="-" THEN
     Dat$=MID$(DATE$,4,2)+"."+LEFT$(DATE$,1)+"-"+RIG
HT$(DATE$,4)
   END IF
   IF LEN(DATE$)=9 AND LEFT$(DATE$,2)<>"-" THEN
     Dat$=MID$(DATE$,4,1)+"."+LEFT$(DATE$,2)+"."+RIG
HT$(DATE$,4)
   END IF
   IF LEN(DATE$)=8 THEN
     Dat$=MID$(DATE$,3,1)+"."+LEFT$(DATE$,1)+"."+RIG
HT$(DATE$,4)
   END IF
RETURN
```

Nach dem Aufruf mit »GOSUB Datum« steht in der Variable »Dat$« das aktuelle Datum im korrekten Format.

(Matthias Specht/M. Jobst/rs)

Selbstgestrickte Icons schützen

Mit Hilfe des CLI-Befehls PROTECT können Sie Ihre selbsterstellten Icons schützen. Dies ist besonders für Basic-Programmierer interessant.

Wenn Sie für Ihr Programm mühsam ein eigenes Icon erstellt haben, dann das Programm ändern und erneut speichern, löscht Amiga-Basic brutal Ihr Icon. Deshalb sollten Sie das Icon beziehungsweise die zugehörige Info-Datei mit diesem Befehl sichern:

```
PROTECT Programmtitel.info R
```

Jetzt hat Amiga-Basic keine Chance mehr. Dieses Icon kann nur noch gelesen werden. *(Rolf-Dieter Schiedrum/rs)*

Unprotect

Eine geschützte Datei oder ein geschütztes Icon (siehe nebenstehender Tip) kann mit »DELETE« nicht mehr gelöscht werden. Dieser Schutz ist sinnvoll, um ein unbeabsichtigtes Löschen von Dateien zu vermeiden. Wenn Sie sich aber doch einmal fest entschlossen haben, eine gesicherte Datei zu verändern, können Sie den Schreibschutz aufheben:

```
PROTECT filename RWED
```

Dieser Befehl erlaubt Ihnen den freien Zugriff auf Ihre Datei.

(Laurent Münster/rs)

Druckeransteuerung in Basic

Die Ansteuerung eines Druckers von Basic aus ist etwas problematisch. Mit Hilfe von LPRINT, LPT1: und PRT: erfolgt die Druckeransteuerung über das »printer.device« des Amiga. Das Device versteht aber nicht die in den Druckerhandbüchern angegebenen Steuercodes, sondern benutzt eigene Kontrollzeichen. Steuert man dagegen einen Drucker, der an der parallelen Schnittstelle angeschlossen ist, direkt über PAR an, so sind die deutschen Umlaute nicht druckbar. Abhilfe schaffen die folgenden kurzen Unterprogramme «txt:« oder »stz:«, die den Drucker je nach Bedarf entweder über PRT oder direkt über PAR ansteuern. Da beim Amiga nur ein Druckerkanal auf einmal geöffnet sein darf, muß immer dafür gesorgt werden, daß jedes Unterprogramm den zuvor geöffneten Kanal schließt:

```
REM Demo für die Druckersteuerung
GOSUB     txt
PRINT#1,  "Dies ist ein Beispiel für die Umlaute:
          äöüÄÖÜ"
GOSUB     stz
PRINT#1,  chr$(27);"W";chr$(1); :REM Breitschrift
          ein
GOSUB     txt
PRINT#1,  "Dies ist Breitschrift äöüÄÖÜ"
CLOSE#1
END
```

```
stz:    REM Steuerzeichen an Drucker
        CLOSE#1 : OPEN "par:" for OUTPUT AS
        #1 :RETURN
txt:    REM Texte mit Umlauten
        CLOSE #1 : OPEN "prt:" for OUTPUT AS #1
        :RETURN
```

(Thomas Küpper/rs)

Scrollen im Listfenster

Das Listfenster des Amiga-Basic läßt sich auf mehrere Arten scrollen:
— Am bekanntesten ist das Scrollen mit Hilfe der Cursortasten.
— Seitenweise läßt sich das Fenster durch gleichzeitiges Drücken der Shift-Taste und der Cursortasten nach oben beziehungsweise unten verschieben.
— Die dritte Methode ist hingegen nicht so bekannt. Das Listfenster können Sie auch mit der Maus bewegen.

Um eine bestimmte Zeile auf dem Bildschirm anzuzeigen, brauchen Sie nur die linke Maustaste drücken und bei gedrückter Taste mit der Maus nach unten oder oben aus dem Fenster zu fahren. Das Listing bewegt sich so lange in die geforderte Richtung, bis Sie den Mauszeiger wieder ins Fenster bringen oder die Maustaste loslassen.

Diese Funktion dient eigentlich zum Markieren eines Blocks, der nachfolgend mit »COPY, CUT und PASTE« bearbeitet werden soll. Um den selektierten Block normal darzustellen, genügt ein Mausklick. Alles ist wie gehabt, nur der Cursor ist genau dort, wo Sie es möchten. *(Heinz Lösch/rs)*

Basic steuert die Hardware

In der AMIGA 8/9/87 steht in der Rubrik »Tips und Tricks«, wie sich die Power-LED mit Hilfe eines Assemblerprogramms ein- und ausschalten läßt. Auch Basicprogrammierer können auf die Hardware zugreifen:

```
POKE 12574721,254 schaltet die LED aus
POKE 12574721,252 schaltet die LED ein
```

Lassen Sie in Ihren Programmen die Leuchtdiode ein paarmal blinken. Verblüffen Sie mit diesem Effekt Ihre Bekannten.
(Daniel Swertz/rs)

RAM-Disk in neuem Kleid

Hatten Sie schon einmal das Bedürfnis, das Icon der RAM-Disk zu ändern, sind Sie daran aber gescheitert? Der folgende Trick hilft sicher: Soll das RAM-Disk-Icon aussehen wie das der Workbench, so fügen Sie einfach in der Startup-Sequenz vor der Zeile

```
PATH RAM:ADD
```

die Sequenz

```
COPY SYS:DISK.INFO RAM:
```

ein. Wollen Sie der RAM-Disk ein individuell gestaltetes Icon verpassen, so entwerfen Sie dieses zunächst wie jedes andere Disketten-Piktogramm mit dem Icon-Editor. Speichern Sie es dann unter dem Namen »DISK1(.info)« auf der Workbench-Diskette ab. Nun muß die an gleicher Stelle wie oben eingefügte Zeile lauten:

```
COPY SYS:DISK1.INFO RAM:DISK.INFO
```

Schon haben Sie beim nächsten Booten ein selbst gewähltes Disk-Icon für die RAM-Disk. *(Christoph Becker/M. Jobst/rs)*

Die »Backups« von Ed und Edit

Wenn man viel mit dem Texteditor Ed oder dem Zeileneditor Edit arbeitet, kann es schon einmal passieren, daß man einen vorher in den Editor geladenen und veränderten Text mit <ESC> <x> überschrieben hat und später dringend auf die alte Version zurückgreifen möchte. Das ist vor allem dann der Fall, wenn man größere Textpassagen aus Versehen gelöscht und dies vor dem Speichern nicht bemerkt hat.

Weitgehend unbekannt ist jedoch, daß die Editoren vor dem Überschreiben eine Sicherheitskopie (Backup) der alten Datei im Unterverzeichnis »t« der Workbenchdiskette (V 33.56) anlegen. Diese Kopien heißen »ed-backup« beziehungsweise »editor-backup«. Mit der Anweisung »if "t/ed-backup"« können Sie eine solchermaßen gerettete Datei in den augenblicklich bearbeiteten Text einfügen. *(Robert Kretzschmar/rs)*

Directory wird umgeleitet

Um mit dem CLI ein Directory zu Papier zu bringen, genügt die Umleitung der Ausgabe auf den Drucker mit »dir >prt:« oder »list >prt:«. Manchmal ist es jedoch von Vorteil, wenn Sie die Ausgabe erst einmal mit »dir >name« in eine Textdatei umleiten, um diese dann mit einem Editor oder dem Notepad zu verändern. So können Sie dem Dateinamen noch ergänzende Informationen, wie etwa Sinn und Zweck des Programms, wie man es gegebenenfalls installiert oder startet, hinzufügen und anschließend die Datei ausdrucken. Sie können diese Technik auch anwenden, wenn nur mit einem Teil der Dateien operiert werden soll: einfach den CLI-Befehl (hier: type) vor die Dateinamen der Textdatei schreiben und das Ganze als Batch-Datei ausführen. *(Lutz Beyert/rs)*

Hilfe für den Floppyspeeder

»Facc« und »Facc II« sind nützliche Programme, die bei vielen Diskettenzugriffen Zeit sparen. Ein Nachteil ist, daß der Cache-Speicher — anders als beim Befehl ADDBUFFERS — von vornherein auf 256 KByte eingestellt ist. Wem dies zuviel ist, der kann die Puffergröße zwar ändern, doch ist diese Arbeit vor jeder Benutzung des »Facc« lästig. Der folgende Patch schafft Abhilfe:
— Facc: Laden Sie mit dem Filemonitor »NewZap« oder »FileZap« (Public Domain) den Sektor 14. An den Cursorpositionen 126 und 127 ist die Puffergröße gespeichert ($0100 = 256). Ändern Sie den Wert nach Ihren Wünschen.
— Facc II: Mit derselben Methode können Sie auch Facc II verändern, jedoch liegt die gesuchte Variable hier in Sektor 1 an der Position $15C und $15D. Arbeiten Sie auf jeden Fall mit einer Sicherheitskopie des Programms. *(Michael Holin/rs)*

Echo mit Effekt

Wer Kommentare in Boot-Sequenzen etwas abwechslungsreicher gestalten will, dem kann nun geholfen werden. Es ist nämlich mit dem ECHO-Befehl des CLI möglich, verschiedene Schriftarten (kursiv, fett, unterstrichen etc.) auf Bildschirm (oder auch Drucker) auszugeben. Man muß dazu das Steuerzeichen »*e«, gefolgt von der Escape-Sequenz des gewünschten Modus, angeben. Die Escape-Sequenzen der einzelnen Modi finden Sie in Anhang C des Benutzerhandbuchs zu Ihrem Amiga. Dazu ein Beispiel:

```
ECHO "*e[4mDas*e[0m ist ja der *e[Wahnsinn*e[0m !!!"
```

bewirkt folgende Bildschirmausgabe:

```
Das ist ja der Wahnsinn !!!
```

(Christian Schmidt/M. Jobst/rs)

Texte drucken mit dem »Ed«

Vielleicht erstellen auch Sie Ihre Quellprogramme mit dem auf der Workbench mitgelieferten Texteditor »Ed«. In keiner Dokumentation findet man allerdings Informationen darüber, wie diese Texte auf den Drucker ausgegeben werden können. Natürlich gibt es eine umständliche Methode: mit »sa "name"« auf Diskette speichern und mit »type >prt: name« ausdrucken. Einfacher ist es, die im Speicher befindliche Datei gleich mit »sa "prt:"« auf den Drucker zu »speichern«. Mit dieser Technik können Sie nach vorheriger Markierung mit »bs« und »be« sowie »wb "prt:"« auch Zeilenbereiche auswählen und diese drucken. Haben Sie Ihren Drucker an der seriellen Schnittstelle angeschlossen, so ersetzen Sie die Gerätenamen »prt:« durch »ser:«. *(Edmund Olt/rs)*

Tips zu Superbase Professional

Die neuen Funktionen von Superbase Professional sind sehr leistungsfähig, wollen aber beherrscht sein. Wenn dazu das Handbuch kleine »Bugs« enthält, kann dies einem die Freude an der Arbeit schon verderben. Hier zeigen wir Ihnen, wie Sie Superbase optimal nutzen.

Einige der neuen Funktionen von Superbase Professional beziehungsweise Superbase 2 werden manchem Anwender — vor allem Einsteigern, die noch nie mit einer professionellen Datenbank gearbeitet haben — Probleme bereiten. Oftmals wird der Sinn und Zweck einer Funktion nicht verstanden, da zunächst nicht klar ist, wie man diese überhaupt einsetzen kann.

Eine dieser neuen Funktionen von Superbase ist die SER-Konstante. Im Handbuch können Sie nur lesen, daß diese SER-Konstante eine automatische Numerierung aller eingegebenen Datensätze bewirkt. Nehmen wir einmal an, Sie möchten mit Superbase Ihre Mitarbeiterdaten verwalten und alle Mitarbeiter mit einer Personalnummer versehen. Um dies zu erreichen, müssen Sie folgendermaßen vorgehen:

Die SER-Funktion

Als erstes erstellen wir eine neue Datei (Bild 1). Nachdem Sie den Namen der Datei — zum Beispiel »sbpeople« — eingegeben haben, erscheint das Ihnen sicherlich schon bekannte Kommunikationsfenster zur Dateidefinition. Jetzt definieren Sie als erstes ein numerisches Feld und geben ihm zum Beispiel den Namen »number«. Jetzt geben Sie zu diesem Feld folgende Konstante ein (Bild 2):

```
SER ("Dateiname")
```

Wenn Sie den oben genannten Namen für Ihre Datei genommen haben, müßte der



Bild 1. Die Definitionsmaske für eine Datei

Eintrag im Konstantenfeld jetzt folgendermaßen lauten:

```
SER ("sbpeople")
```

Im Superbase-Handbuch wurden leider die Anführungszeichen zwischen den Klammern vergessen. Sollten Sie es also nicht geschafft haben, eine SER-Konstante zu definieren, wissen Sie jetzt, wo der Fehler lag.

Automatische Numerierung

Nun müssen Sie nur noch die anderen Felder, die zu einer solchen Datenbank gehören, definieren. Dies könnten Felder sein wie Abteilung, Titel etc. Bei der Eingabe Ihrer Daten wird jetzt der erste Mitarbeiter die Nummer eins erhalten, der zweite die Nummer zwei etc.

Wenn Sie später einen neuen Mitarbeiter in Ihr Team aufnehmen wollen, müssen Sie nur einen neuen Datensatz öffnen, und schon hat Ihr neuer Mann seine Nummer. Sollte jemand ausscheiden, brauchen Sie nur den entsprechenden Datensatz zu löschen. Da die SER-Funktion als Konstante definiert ist, wird die Datei nicht neu durchnumeriert, sondern jeder Datensatz behält seine Nummer.

In diesem Zusammenhang ist es vielleicht noch erwähnenswert, daß eine Datei, bei der die SER-Funktion genutzt wurde, nicht reorganisiert werden kann. Sollten Sie es doch versuchen, erscheint eine Fehlermeldung wie in Bild 3 dargestellt.

Das Ganze mag Ihnen zunächst sehr unsinnig erscheinen, die Erklärung ist aber ganz einfach. Nehmen wir an, Sie hätten verschiedene Datensätze gelöscht und Superbase würde einen solchen Reorganisationslauf durchführen, dann wäre die ganze Datei neu durchnumeriert. Können

Reorganisation nicht möglich

Sie sich das Durcheinander vorstellen, wenn Mitarbeiter XY plötzlich die Personalnummer 12 besitzt und vor der Reorganisation war es die Nummer 4. Sollten Sie dennoch den



Bild 2. So geben Sie die SER-Funktion ein

Wunsch verspüren, Ihre Artikeldatei zu reorganisieren, ist dies nur über die Import/Export-Funktion möglich. Exportieren Sie zuerst alle Daten Ihrer Datei, mit Ausnahme des Feldes, auf die Sie die SER-Funktion angewandt haben. Danach erstellen Sie eine neue Datei nach dem Vorbild der alten und importieren die Daten wieder.

Achtung: Wenn Sie Schwierigkeiten mit der Import/Export-Funktion haben, empfehlen wir Ihnen, das entsprechende Kapitel im Handbuch noch einmal genau durchzulesen. Versuchen Sie nicht, nach dem Exportieren der Daten aus Ihrer Datei alle Datensätze zu löschen und dann die ASCII-Datei mit Ihren Daten wieder zu importieren. Die SER-Konstante einer Datei können Sie nachträglich nicht mehr löschen. Wenn in Ihrer Datei schon 70 Datensätze gespeichert waren und Sie importieren dieselbe ASCII-Datei, die Sie vorher schon exportiert haben, erhält der erste eingelesene Datensatz die Nummer 71.

Mehrfacheintragungen

Ebenfalls eine neue Funktion ist die Möglichkeit der Mehrfacheintragungen in Textfeldern (Bild 4). In Textfeldern mit Mehrfacheintragungen wird immer nur der erste Eintrag angezeigt, alle anderen Einträge sehen Sie nur, wenn Sie die Tasten <CTRL P> oder <CTRL N> innerhalb dieses Textfeldes betätigen. Im Superbase-Handbuch ist ein schönes Beispiel abgedruckt, das Ihnen zeigt, wie man eine ganze Adresse in einem Feld unterbringt.

Toll, jetzt kann ich ja viel Platz sparen, werden Sie nun vielleicht denken. Früher brauchte man vier bis fünf Datenfelder für eine Adresse und ab sofort benötigt man nur noch ein Feld.

Spätestens wenn Sie versuchen, Ihre Adressen auf Etiketten auszudrucken oder einen Serienbrief zu schreiben, werden Sie feststellen müssen, daß dies mit Eintragungen aus Mehrfachtextfeldern nicht möglich ist. Die einzige Möglichkeit, die Mehrfacheintragungen auszuwerten, besteht mit der Filter-Funktion von Superbase. Wenn Sie zum Beispiel alle Meier in Ihrem zweiten Mehrfachtextfeld suchen,



Bild 3. Die Reorganisation einer mit der SER-Funktion bearbeiteten Datei ist nicht möglich

müssen Sie folgende Syntax verwenden:

```
FELDNAME(1) LIKE "Meier"
```

Beachten Sie, daß Sie beim Anzeigen Ihrer Filterliste kein Feld mit dem Eintrag »Meier« sehen werden, da immer nur der erste Eintrag angezeigt wird. Leider ist auch in diesem Fall das im Handbuch angegebene Beispiel nicht ganz korrekt. Die Syntax FELDNAME(2) bezieht sich nicht — wie im Handbuch erläutert — auf den zweiten Mehrfacheintrag, sondern auf den dritten. Dieses Beispiel sollte Ihnen zeigen, daß Sie die Mehrfacheintragungen in Textfeldern nur als zusätzliche Informationen verwenden sollten. Sicher waren Sie schon in der Situation, daß Sie reihenweise Daten eingeben mußten und es Ihnen nach einiger Zeit sicher auf die Nerven gegangen ist, daß Sie jeden eingegebenen Datensatz einzeln speichern müssen.

Mit den neuen Superbase-Versionen wird dieses Problem auch gelöst. Sie haben die Möglichkeit, im Datensatzmenü den sogenannten Stapelmodus einzuschalten. Sobald dieser aktiviert ist, speichert Superbase die eingegebenen Datensätze nicht sofort, sondern das Programm behält Ihre Daten im Speicher, bis Sie den Stapelmodus wieder ausschalten. Erst dann werden alle Daten auf Diskette gesichert.

Es wird gestapelt

Es ist möglich, daß Superbase auch während der Stapelmodus eingeschaltet ist, auf die Diskette zugreift. Das entspricht zwar nicht ganz den Angaben im Handbuch, aber Sie brauchen deswegen keine Angst zu haben. Alle eingegebenen Daten werden korrekt gespeichert. Auch wenn der Stapelmodus eingeschaltet ist, müssen Sie nach jedem eingegebenen Datensatz die Tasten <Amiga-rechts S> und <Amiga-rechts N> betätigen oder im Menü Datensatz die Option »Speichern« und »Neu« aufrufen.

Bei der Eingabe von großen Datenmengen sollten Sie den Stapelmodus immer wieder einmal abschalten, um zu speichern. Sollte plötzlich Strom ausfallen oder Ihr Rechner es vorziehen, zu meditieren, gehen nur wenige Daten verloren.

Einsatz einer RAM-Disk

Wenn Sie zu den Glücklichen gehören, deren Amiga mit genügend RAM ausgestattet ist, können Sie natürlich Ihre Dateien in einer RAM-Disk bearbeiten. Dazu müssen Sie im File-Requester lediglich vor dem Dateinamen »RAM:« eingeben. Bei der Bearbeitung einer Datei in der RAM-Disk wird deutlich, wie schnell der Amiga mit Superbase wirklich ist. Sobald Sie Ihre RAM-Datei wieder schließen, fragt Superbase automatisch, ob die Datei wieder auf die Diskette zurückgeschrieben werden soll.

In loser Folge finden Sie in den nächsten Amiga-Sonderheften weitere Tips & Tricks zu Superbase. Das nächste Mal werden wir uns mit Lokup- und der Tenär-Funktion beschäftigen. *(Dietmar Inäbnit/sk)*



Bild 4. Im Feld »Eintragungen« lassen sich Mehrfacheintragungen festlegen

Schach der Langeweile!

Strategen aufgepaßt! Mit Shogun präsentieren wir Ihnen ein ganz besonderes Programm: ein altes japanisches Brettspiel, das dem Schach sehr ähnlich ist.

Es ist schwül, die Luft ist stickig. Kein Laut durchdringt die trügerische Stille. Das Gefecht ist an einem entscheidenden Punkt angelangt. Die Redakteure fixieren sich aus den Augenwinkeln. Wer macht den ersten Fehler? Plötzlich gellt ein markerschütternder Schrei. Verdammt, das war einer zuviel, denke ich und überlege mir bereits eine neue Angriffstaktik, um meinem Gegner endgültig den Garaus zu machen.

Solch fesselnde Szenen können auch Sie erleben. Sie benötigen lediglich einen Amiga, Shogun und einen Mitspieler als Opfer. Haben Sie das alles? Gut, dann kann es ja losgehen.

Shogun wird von zwei Spielern gespielt. Das Ziel des Spiels ist es den gegnerischen Shogun mattzusetzen, oder alle seine Truppen aufzureiben. Die Spielregeln sind schnell erklärt: Das Spielfeld besteht aus einem 8 x 8 Felder großen Bereich, in dem Sie Ihre Spielsteine bewegen können. Jeder der acht Steine ist mit einer Zahl von 1 bis 4 gekennzeichnet, die sich nach jedem Zug ändert. Die Steine sind anfangs oben und unten in den Grundreihen aufgestellt (Bild 1). Die Anfangsaufteilung ist jeweils verschieden, die Zahlen auf den Spielfiguren besitzen also anfangs immer verschiedene Werte.

Der Stein mit der gelben Markierung ist der Shogun. Diese, dem König beim Schach entsprechende Figur, kann lediglich die Zahlen 1 und 2 annehmen. Die auf den Steinen aktuell angezeigte Zahl muß gezogen werden, wobei man sich nur waagerecht und senkrecht, nie diagonal fortbewegen kann. Die Zugrichtung kann vorwärts, rückwärts oder seitwärts sein. Bei jedem Zug darf man einmal(!) einen Richtungswechsel im rechten Winkel vornehmen. Trifft man bei einem Zug genau auf ein Feld, das bereits von einem gegnerischen Stein besetzt ist, so wird der Gegner aus dem Spiel genommen. Liegt ein Stein, egal ob ein eigener oder einer des Gegners, im Weg, so darf dieser nicht aus dem Spiel genommen und auch nicht übersprungen werden.

Ein Shogun ist bedroht, wenn eine gegnerische Figur diesen beim nächsten Zug schlagen könnte. Ihr Amiga meldet dies mit einem Warnton und der Textausgabe »Ihr shogun ist bedroht«. Der Shogun muß dann geschützt werden, indem
— er auf ein unbedrohtes Feld gesetzt wird,
— er den bedrohenden Stein schlägt, wenn dieser nicht von einer anderen gegnerischen Figur gedeckt ist,
— ein anderer Stein die generische Figur schlägt, oder
— eine andere Figur so zwischen Angreifer und Shogun gesetzt wird, daß der Bedroher nicht mehr auf das Feld des Shogun zugreifen kann.

Sekt oder Selters - das Spielende

Es gibt zwei Wege, das Spiel zu beenden: Erstens, den gegnerischen Shogun mattzusetzen, daß heißt ihn so zu bedrohen, daß er sich nirgendwohin mehr retten kann. Und zweitens, alle Soldaten des Feindes, bis auf einen zu vernichten (Bild 2).

Die Spielregeln hören sich eventuell etwas kompliziert an, spätestens nach den ersten Spielen sind Sie mit den Regeln jedoch bestens vertraut. Dann werden Sie viele listige Spielzüge entdecken, die dem Gegner den Angstschweiß auf die Stirn treiben. Ein Hinweis noch: die Wertigkeit der Spielfiguren ist auf den gleichen Feldern in einem Spiel immer gleichbleibend. Das Schema, nach dem die Figuren ihre Wertigkeit verändern, möchten wir Ihnen an dieser Stelle nicht verraten. Vielleicht finden Sie ja die Logik, die dahintersteht, heraus (kleiner Tip: auch das Spielfeld verändert seine Lage in jedem Spiel und es gibt Kategorien von Spielsteinen).

Insgesamt gibt es drei verschiedene Spielsteinsorten. Jedem Stein ist eine der Bezeichnungen a, b, c, oder d in einer festgelegten Reihenfolge zugeteilt:

a	b	c	d	Spielfigur
1	1	2	2	Shogun
1	3	2	4	Typ 1
1	2	4	3	Typ 2

Ein Shogun kann also die Zahlen 1 oder 2 zeigen, die anderern Steine die Zahlen 1 bis 4. Jedem der 64 Spielfelder ist stets eine der entsprechenden Bezeichnungen »a« bis »d« zugeordnet.

Jetzt gibt's keine Hinweise mehr, sonst sind Sie am Ende Ihren Gegnern vielleicht zu hoch überlegen, und das Spiel wird für die Gegner frustrierend.

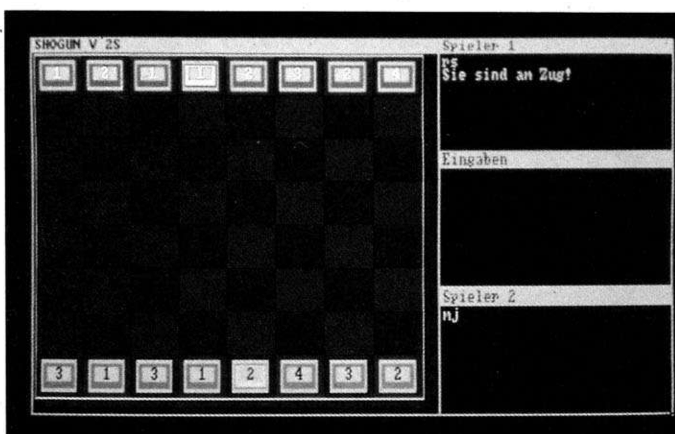


Bild 1. Die Anfangsszene von Shogun, noch sind alle Spielsteine auf dem Brett...

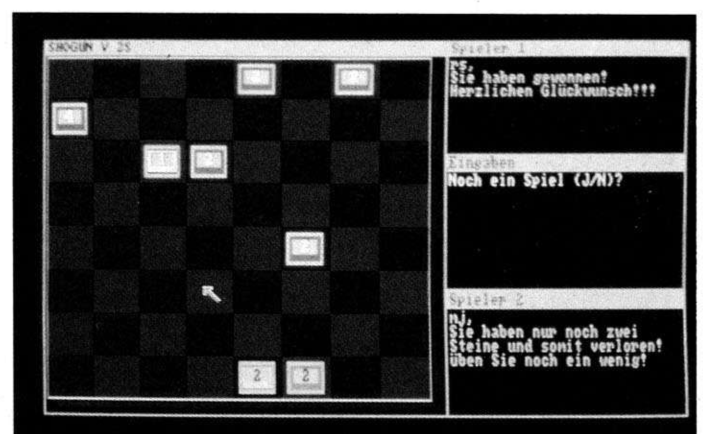


Bild 2. ...am Ende ist der Spieler Weiß geschlagen, da er neben dem Shogun nur noch eine weitere Figur besitzt.

Hinweise zum Programm

Geben Sie Listing 1 mit dem Checksummer (Seite 159) ein und speichern es unter dem Namen »Shogun« auf eine Diskette, auf der auch das Amiga-Basic enthalten ist. Darauf erstellen Sie ein Info-File mit demselben Namen und brauchen dieses nur noch zu laden.

Nach dem Starten des Programms erscheint das Titelbild. Sie können die Spielregeln noch einmal lesen, oder direkt ins Spiel verzweigen (<n> eingeben). In diesem Fall erscheint die Meldung »Einen Moment bitte...«, kurz darauf wird das Feld aufgebaut.

Auf diesem Bildschirm sind zusätzlich drei weitere Fenster zu sehen: Das Fenster »Eingaben« dient zur Eingabe von Daten bei Programmabfragen. Die Fenster »Spieler 1« und »Spieler 2« sind reine Ausgabefenster, hier erscheinen im Programmverlauf verschiedene Informationen.

Geben Sie nun im Eingabefenster die Namen für Spieler 1 und 2 an, indem Sie die Namen eintippen und dies mit <RETURN> bestätigen. Anschließend erscheint im Feld »Spieler 1« die Aufforderung »Sie sind am Zug«.

Dieser fährt nun mit dem Mauszeiger auf die Figur, die er bewegen möchte. Er klickt sie mit der linken Maustaste an, anschließend fährt er auf das Zielfeld und drückt erneut die linke Maustaste. Der Zug wird ausgeführt und der zweite Spieler ist an der Reihe.

War der angegebene Zug nicht regelgemäß, erscheint eine Fehlermeldung. Sie müssen sich dann ein anderes Zielfeld aussuchen. Entscheiden Sie sich nach dem Anklicken der gewünschten Figur doch für eine andere, klicken Sie die erste erneut an und dann die jetzt gewünschte.

Die Überprüfung, ob ein Shogun »matt« gesetzt ist, übernimmt das Programm. Dies kann bei seltenen, ungünstigen Situationen einige Sekunden dauern. Dies wird mit einer Meldung bekanntgegeben.

Ist das Spiel beendet, fragt Sie das Programm, ob eine weitere Partie gewünscht wird. Geben Sie daraufhin <j> oder <n> ein. Werden die Eingaben nicht angenommen, kann es daran liegen, daß das Eingabefenster im Moment nicht das aktuelle ist. Sie erkennen das an der »Geisterschrift«. Ein kurzer Klick auf die linke Maustaste genügt dann, der Mauszeiger muß sich dabei im Eingabefenster befinden.

Nun aber los: hinsetzen, abtippen, spielen.

(Frank Garnath/rs)

Programmname:	Shogun
Computer:	Amiga 500, 1000, 2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programm : Shogun
-----------------

 1 HOO '  ****************************************
 2 WD  '  * Programm SHOGUN V 2S                 *
 3 mV  '  * von Frank Garnath    1987/1988       *
 4 KR  '  ****************************************
 5 zb3    CLEAR ,,30000
 6 Af     DEFINT f,m-s,w,z
 7 TW     SCREEN 2,640,215,3,2
 8 fn     WINDOW 7,"",(0,0)-(631,190),0,2
 9 TV     WINDOW 8,"",(170,72)-(473,127),0,2:LINE (0,0)-(303,65),1
          ,bf
10 ot     WINDOW 6,"",(162,68)-(465,123),0,2
11 IN     PALETTE 0,.9,.9,1
12 3T     FOR x=1 TO 7
13 Lp5      PALETTE x,x/10-.1,x/10-.1,x/10+.1
14 9e3    NEXT x
15 OI     COLOR 1,0
16 lr     LOCATE 1,12:PRINT" S H O G U N "
17 TN     LOCATE 3,9:PRINT"von  Frank Garnath"
18 lN     LOCATE 5,9:PRINT"Lübeck, 1987/1988"
19 kP     LOCATE 7,1:PRINT"Möchten Sie die Anleitung lesen (j/n)?"

20 yk     farbe=1:add=1
21 Dx     WHILE i$<>"j" AND i$<>"n"
22 PY5      i$=INKEY$:RANDOMIZE TIMER
23 MK       WINDOW OUTPUT 7
24 Qp       LINE (x,y)-(631-x,200-y),farbe,b:LINE (x+1,y)-(630-x,2
            00-y),farbe,b
25 dV       farbe=farbe+add
26 yd       IF farbe=7 THEN add=-1
27 iY       IF farbe=1 THEN add=1
28 Lt       IF z=0 AND x<631 THEN x=x+2:y=y+1 ELSE z=1:x=x-2:y=y-
            1:IF x<0 THEN z=0
29 ui3    WEND
30 28     CLS
31 11     WINDOW CLOSE 6
32 Cy     WINDOW CLOSE 8
33 iW     IF i$="j" THEN CALL Anleitung
34 JA     GOSUB Dunkel:PALETTE 2,0,0,0:COLOR 2,0
35 1B     CLS:LOCATE 12,30:PRINT"Einen Moment bitte!"
36 qn     GOSUB Hell
37 jz0 Voreinstellungen:
38 zN3    DIM SHARED f(63), wert(63), ZugR(7,15), ZugW(7,15)
39 WQ1  NeuesSpiel:
40 ok3    RANDOMIZE TIMER:v=INT(RND*4)
41 2K     IF v=0 OR v=2 THEN RESTORE Daten0:f1=2:f2=3
42 vk     IF v=1 OR v=3 THEN RESTORE Daten90:f1=3:f2=2
43 MU     IF v>1 THEN a=1:b=0:c=-1 ELSE a=0:b=1:c=1
44 8s     FOR y=a TO b STEP c
45 AD5      IF v>1 THEN d=15+y*32:e=y*32:f=-1 ELSE d=y*32:e=15+y*
            32:f=1
46 bY       FOR x=d TO e STEP f
47 Et7        READ wert(x):wert(x+16)=wert(x)
48 hC5      NEXT x
49 kG3    NEXT y
50 Wn     FOR y=0 TO 3
51 Gw5      FOR x=y*16 TO 6+y*16 STEP 2
52 NK7        f(x)=f1:f(x+1)=f2:f(x+8)=f2:f(x+9)=f1
53 mH5      NEXT x
54 pL3    NEXT y
55 4D     RANDOMIZE TIMER
56 i7     FOR x=0 TO 7
57 Nq5      prx(x)=1+x*48:pry(x)=1
58 OF       pwx(x)=prx(x):pwy(x)=pry(x)+168
59 Jr       zr(x)=wert(x):zw(x)=wert(x+56)
60 Ze       v=RND
61 71       IF v<.5 THEN zar(x,1)=1:zar(x,2)=3:zar(x,3)=2:zar(x,4
            )=4
62 Em       IF v>=.5 THEN zar(x,1)=1:zar(x,2)=2:zar(x,3)=4:zar(x,
            4)=3
63 ch       v=RND
64 IW       IF v<.5 THEN zaw(x,1)=1:zaw(x,2)=3:zaw(x,3)=2:zaw(x,4
            )=4
65 JC       IF v>=.5 THEN zaw(x,1)=1:zaw(x,2)=2:zaw(x,3)=4:zaw(x,
            4)=3
66 zU3    NEXT x
67 XI     zar(3,1)=1:zar(3,2)=1:zar(3,3)=2:zar(3,4)=2
68 Ap     zaw(4,1)=1:zaw(4,2)=1:zaw(4,3)=2:zaw(4,4)=2
69 M3     IF ns=1 THEN Spielfeld
70 9n     FOR y=.75 TO .02 STEP -.01
71 fy5      FOR x=0 TO 1
72 TV7        PALETTE x,y,y+.22,y+.25
73 6b5      NEXT x
74 9f3    NEXT y
75 7Y     FOR x=2 TO 7
76 bF5      PALETTE x,.02,.24,.27
77 Af3    NEXT x
78 rc     WINDOW CLOSE 7
79 5u     WINDOW 2,"SHOGUN V 2S",(0,0)-(397,200),0,2
80 9Z     FOR x=0 TO 8
81 oS5      LINE (10,5)-(387+x,194+INT(x/2)),2,b
82 PU3    NEXT
83 zN0 Spielfeld:
84 VL3    WINDOW OUTPUT 2:LOCATE 2,1
85 QS     LINE (0,0)-(385,193),5,b
86 Et     FOR y=1 TO 163 STEP 48
87 J65      FOR x=1 TO 317 STEP 96
88 KM7        LINE (x,y)-(x+47,y+23),f1,bf
89 Px         LINE (x+48,y)-(x+95,y+23),f2,bf
90 wU         LINE (x,y+24)-(x+47,y+47),f2,bf
91 KN         LINE (x+48,y+24)-(x+95,y+47),f1,bf
92 Pu5      NEXT x
93 Sy3    NEXT y
94 K80 Figuren:
95 Lk3    FOR x=0 TO 7
96 Ip5      LINE (6+x*48,4)-(41+x*48,21),4,bf
97 Wt       LINE (6+x*48,172)-(41+x*48,189),1,bf
98 Cl       LINE (8+x*48,6)-(39+x*48,19),6,b
```

```
oX      LINE (8+x*48,174)-(39+x*48,187),7,b
of      IF x<>3 THEN PAINT (20+x*48,12),6
wu      IF x<>4 THEN PAINT (20+x*48,179),7
Z43    NEXT x
1U     PAINT (164,12),5,6:PAINT (212,179),5,7
xN     COLOR 5,4
Vu     FOR x=0 TO 7
Pr5     PRINT PTAB(12+x*48)zar(x,zr(x));
e93    NEXT x
Ow     COLOR 2,1:LOCATE 23,1
Zy     FOR x=0 TO 7
xZ5     PRINT PTAB(12+x*48)zaw(x,zw(x));
iD3    NEXT x
B10 Eingaben:
bu3    WINDOW 4,"Spieler 1",(404,0)-(631,54),0,2
jL     WINDOW 5,"Spieler 2",(404,143)-(631,200),0,2
Xe     WINDOW 3,"Eingaben",(404,67)-(631,129),0,2
Rv     PALETTE 0,.02,.24,.27
j1     PALETTE 1,1,1,1
di     PALETTE 2,0,0,0
tM     PALETTE 3,.33,.33,.33
an     PALETTE 4,.93,.2,0
IA     PALETTE 5,1,1,.13
1W     PALETTE 6,.53,.15,0
oF     PALETTE 7,.63,.63,.63
xJ     COLOR 5,0
dy     FOR x=1 TO 2
D65     PRINT "Name Spieler",x:INPUT na$(x)
Ci      BEEP
zU3    NEXT x
00     WINDOW OUTPUT 4:COLOR 4,0:PRINT na$(1)
ys     WINDOW OUTPUT 5:COLOR 1,0:PRINT na$(2)
ro     WINDOW OUTPUT 3:COLOR 5,0
t5     CLS:sp=1
eR0 naechsterSpieler:
yk3    IF sp=0 THEN sp=1 ELSE sp=0
tu     SteinNr=-2
bF0 Spielbeginn:
Mc3    IF mattpr=1 THEN Mattpruef
VO     WINDOW OUTPUT 4+sp:COLOR 4-sp*3,0:CLS:PRINT na$(1+sp)
bu     PRINT"Sie sind am Zug!":PRINT p$
13     WINDOW OUTPUT 2
wy1  CheckMouse1:
nx3    WHILE MOUSE(0)<1
kY     WEND
J3     mx=MOUSE(3):my=MOUSE(4)
0Q     mx=INT(mx/48)*48+1:my=INT(my/24)*24+1
d3     IF sx=0 THEN sx=1:sy=1
vC     IF px=0 THEN px=1:py=1
2T     LINE (sx,sy)-(sx+47,sy+23),f(sx/48+sy/3),b
qi     LINE (px,py)-(px+47,py+23),f(px/48+py/3),b
Ed     FOR x=0 TO 7
S15     IF prx(x)=mx AND pry(x)=my THEN m=x:GOTO MarkSetz
Ns3    NEXT x
Y9     GOTO CheckMouse1
Yu1  MarkSetz:
tS3    BEEP:LINE (mx,my)-(mx+47,my+23),5,b
DG1  CheckMouse2:
2K3    WINDOW OUTPUT 2
3D     WHILE MOUSE(0)<1
Oo     WEND
95     sx=MOUSE(3):sy=MOUSE(4)
6Z     IF sx>383 OR sy>191 THEN CheckMouse2
j3     sx=INT(sx/48)*48+1:sy=INT(sy/24)*24+1
rk     IF sx=prx(m) AND sy=pry(m) THEN LINE (sx,sy)-(sx+47,sy+2
       3),f(sx/48+sy/3),b:GOTO Spielbeginn
1A     LINE (sx,sy)-(sx+47,sy+23),5,b
Oa1  EinsprungMattpruef:
713    schrittzahl=ABS(INT((prx(m)-sx)/48))+ABS(INT((pry(m)-sy)
       /24))
Dp     IF schrittzahl<>zar(m,zr(m)) THEN
9J5     fz=1:fz1$="Sie müssen":fz2$="Feld(er) ziehen!"
05      GOTO FalscherZug
gZ3    END IF
Zy     FOR x=0 TO 7
OP5     IF sx=prx(x) AND sy=pry(x) THEN
C77      IF schwpr=1 THEN Sprungpruef
Wf       fz1$="Sie können nicht Ihren":fz2$="eigenen Stein sc
         hlagen!"
6B       GOTO FalscherZug
mf5     END IF
mH3    NEXT x
aa1  Sprungpruef:
YJ3    IF zar(m,zr(m))=1 THEN AllgShogunpruef1
eL     weg1=0:weg2=0
Ze     IF sx=prx(m) THEN ZugpruefY
ij     IF sy=pry(m) THEN ZugpruefX
8t     IF sx<prx(m) THEN stpx=48
2M     IF sx>prx(m) THEN stpx=-48
qn     IF sy<pry(m) THEN stpy=24
yK     IF sy>pry(m) THEN stpy=-24
I2     FOR n=sx+stpx TO prx(m) STEP stpx
yx5     FOR o=0 TO 7
wd7      IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)=
         sy) THEN weg1=1
te       IF (prx(o)=n-stpx AND pry(o)=pry(m)) OR (pwx(o)=n-st
         px AND pwy(o)=pry(m)) THEN weg2=1
i45     NEXT o
h23    NEXT n
5k     FOR n=sy+stpy TO pry(m)-stpy STEP stpy
435     FOR o=0 TO 7
007      IF (pry(o)=n AND prx(o)=prx(m)) OR (pwy(o)=n AND pwx
         (o)=prx(m)) THEN weg1=1
1c       IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)=
         sx) THEN weg2=1
oA5     NEXT o
n83    NEXT n
66     IF weg1=1 AND weg2=1 THEN FZug
Gr     GOTO AllgShogunpruef1
p71  ZugpruefX:
OH3    IF sx<prx(m) THEN stp=48
dD     IF sx>prx(m) THEN stp=-48
7f     FOR n=sx+stp TO prx(m)-stp STEP stp
FE5     FOR o=0 TO 7
jS7      IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o)=
         sy) THEN FZug
yK5     NEXT o
xI3    NEXT n
P0     GOTO AllgShogunpruef1
3M1  ZugpruefY:
s03    IF sy<pry(m) THEN stp=24
PA     IF sy>pry(m) THEN stp=-24
Pz     FOR n=sy+stp TO pry(m)-stp STEP stp
ON5     FOR o=0 TO 7
Rn7      IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o)=
         sx) THEN
Fd1  FZug:
059        fz1$="Sie dürfen keinen Stein":fz2$="überspringen!
           ":fz=0
ns         GOTO FalscherZug
TM7      END IF
BX5     NEXT o
AV3    NEXT n
CEO AllgShogunpruef1:
be3    spp=sp:sxx=sx:syy=sy
um     zz=zr(m):zr(m)=wert(INT(sx/48+sy/3))
wM     pxm=prx(m):pym=pry(m):prx(m)=sx:pry(m)=sy
AZ     z=-1
JMO AllgShogunpruef2:
TF3    IF sp=0 THEN
2q5     sx=prx(3):sy=pry(3)
Wv      FOR x=0 TO 7
hX7      IF spp=0 AND pwx(x)=sxx AND pwy(x)=syy THEN pxmem=pw
         x(x):pwx(x)=0:pymem=pwy(x):pwy(x)=0:z=x:pm=1
fA5     NEXT x
Fo      GOSUB ShogunBedroht
Zv      IF spp=0 THEN sp=1:GOSUB WerteTauschen:GOTO AllgShogun
        pruef2 ELSE GOTO Pruefende
073    ELSE
H75     sx=prx(4):sy=pry(4)
d2      FOR x=0 TO 7
sJ7      IF spp=1 AND pwx(x)=sxx AND pwy(x)=syy THEN pxmem=pw
         x(x):pwx(x)=0:pymem=pwy(x):pwy(x)=0:z=x:pm=1
mH5     NEXT x
Mv      GOSUB ShogunBedroht
11      IF spp=1 THEN sp=0:GOSUB WerteTauschen:GOTO AllgShogun
        pruef2
qJ3    END IF
jh1  Pruefende:
mu3    sx=sxx:sy=syy:sp=spp
Po     GOSUB WerteTauschen
mn0 Steinsetz:
Tf3    IF spm=1 AND pm=1 THEN pm=0:IF z<>-1 THEN pwx(z)=pxmem
       :pwy(z)=pymem
fM     IF spm=1 THEN MattEnde
```

Listing 1. »Shogun« bitte mit dem Checksummer (Seite 159) eingeben

```
249 pF    WINDOW OUTPUT 4+sp:CLS:PRINT na$(1+sp)
250 Xp    WINDOW OUTPUT 2
251 Ci    BEEP
252 Te    LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23),f
          (prx(3+sp)/48+pry(3+sp)/3),b
253 pG    LINE (mx,my)-(mx+47,my+23),f(mx/48+my/3),bf
254 3D    prx(m)=sx:pry(m)=sy:zr(m)=wert(INT(prx(m)/48+pry(m)/3))
255 12    LINE (prx(m)+5,pry(m)+3)-(prx(m)+40,pry(m)+20),4-sp*3,bf

256 H2    LINE (prx(m)+8,pry(m)+5)-(prx(m)+38,pry(m)+18),6+sp,b
257 DQ    IF m=3+sp THEN fa=5 ELSE fa=6+sp
258 To    PAINT (prx(m)+20,pry(m)+12),fa,6+sp
259 dl    LINE (prx(m),pry(m))-(prx(m)+47,pry(m)+23),f(prx(m)/48+p
          ry(m)/3),b
260 Br    LOCATE pry(m)/8+2,1:COLOR 5-sp*3,4-sp*3
261 x6    PRINT PTAB(prx(m)+11);zar(m,zr(m))
262 g4    IF sp=sb THEN p$=""
263 op    IF z<>-1 THEN GOSUB FigurGeschlagen
264 i7    GOSUB WerteTauschen
265 CY    GOTO naechsterSpieler
266 ga0 Summen:
267 bA3   zzz=zahl-1
268 Gy    sumX(1,1)=0:sumX(1,2)=0:sumX(1,3)=-zahl*48:sumX(1,4)=zah
          l*48:sumY(1,1)=-zahl*24:sumY(1,2)=zahl*24:sumY(1,3)=0:su
          mY(1,4)=0
269 nY    IF Pruefstufe=1 THEN RETURN
270 Po    sumX(2,1)=-48:sumX(2,2)=48:sumX(2,3)=-48:sumX(2,4)=48:su
          mY(2,1)=-zzz*24:sumY(2,2)=zzz*24:sumY(2,3)=zzz*24:sumY(2
          ,4)=-zzz*24
271 vc    IF Pruefstufe=2 THEN RETURN
272 GC    sumX(3,1)=-zzz*48:sumX(3,2)=zzz*48:sumX(3,3)=-zzz*48:sum
          X(3,4)=zzz*48:sumY(3,1)=-24:sumY(3,2)=24:sumY(3,3)=24:su
          mY(3,4)=-24
273 3g    IF Pruefstufe=3 THEN RETURN
274 Ud    sumX(4,1)=-96:sumX(4,2)=96:sumX(4,3)=-96:sumX(4,4)=96:su
          mY(4,1)=-48:sumY(4,2)=48:sumY(4,3)=48:sumY(4,4)=-48
275 nP    RETURN
276 Mw0 Mattpruef:
277 z63   SteinNr=SteinNr+1:m=SteinNr
278 i3    IF SteinNr=-1 THEN IF sp=0 THEN m=3 ELSE m=4
279 2z    IF SteinNr=0 THEN WINDOW OUTPUT 4+sp:PRINT"Einen Augenbl
          ick...":WINDOW OUTPUT 2
280 Bl    IF (sp=0 AND SteinNr=3) OR (sp=1 AND SteinNr=4) THEN Mat
          tpruef
281 Ef    IF SteinNr>7 THEN VerlorenGewonnen
282 vj    IF prx(m)=0 THEN Mattpruef
283 zg    Pruefstufe=1:pr=0
284 Vv    zahl=zar(m,zr(m))
285 PB    mx=prx(m):my=pry(m):zrzahl=zr(m)
286 7u1 EinsprungMEnde:
287 413   spm=1:MEnde=0
288 eM    GOSUB Summen
289 Ko1 Pruefstufen:
290 fC3   pr=pr+1
291 dB    sx=prx(m)+sumX(Pruefstufe,pr):sy=pry(m)+sumY(Pruefstufe,
          pr):IF sx<1 OR sy<1 OR sx>337 OR sy>169 THEN MEnde=1
          :GOTO MattEnde
292 Rq    IF pr>4 AND zahl=Pruefstufe AND SteinNr=7 THEN Verloren
          Gewonnen
293 cu    IF pr>4 AND zahl=Pruefstufe THEN naechstStein=1:GOSUB M
          attEnde:GOTO Mattpruef
294 77    IF pr>4 AND zahl>Pruefstufe THEN Pruefstufe=Pruefstufe
          +1:MEnde=1:pr=0:GOTO MattEnde
295 rt    IF Pruefstufe=5 THEN naechstStein=1:MEnde=0:GOSUB MattEn
          de:GOTO Mattpruef
296 5q    GOTO EinsprungMattpruef
297 WV1 MattEnde:
298 xX3   spm=0:sx=0:sy=0:prx(m)=mx:pry(m)=my:zr(m)=zrzahl
299 QY    IF MEnde=1 THEN EinsprungMEnde
300 ec    IF naechstStein=1 THEN naechstStein=0:RETURN
301 Nc    mattpr=0:SteinNr=-2
302 hu    GOTO Spielbeginn
303 eV0 ShogunBedroht:
304 c53   FOR w=0 TO 7
305 i75     shfeld=ABS((pwx(w)-sx)/48)+ABS((pwy(w)-sy)/24)
306 uR      IF shfeld=zaw(w,zw(w)) THEN
307 q67       px=pwx(w):py=pwy(w)
308 4L        IF px<>0 AND shfeld=1 THEN FZug2
309 9A        IF px<>0 THEN GOSUB Shogunpruef
310 wp5     END IF
311 u03   NEXT w
312 00    RETURN
313 xD0 Shogunpruef:
314 oV3   weg1=0:weg2=0
315 IA    IF sx=px THEN ShogunpruefY
316 RF    IF sy=py THEN ShogunpruefX
317 MM    IF sx<px THEN stpx=48
318 nb    IF sx>px THEN stpx=-48
319 IK    IF sy<py THEN stpy=24
320 nT    IF sy>py THEN stpy=-24
321 UI    FOR n=sx+stpx TO px STEP stpx
322 875     FOR o=0 TO 7
323 vH7       IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
324 se9         IF pwx(o)=n AND pwy(o)=sy THEN weg1=1
325 5u          IF pwx(o)=n-stpx AND pwy(o)=py THEN weg2=1
326 ra7       ELSE
327 Ar9         IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o
            )=sy) THEN weg1=1
328 7F          IF (prx(o)=n-stpx AND pry(o)=py) OR (pwx(o)=n-stpx
             AND pwy(o)=py) THEN weg2=1
329 F87       END IF
330 xJ5     NEXT o
331 wH3   NEXT n
332 Ik    FOR n=sy+stpy TO py-stpy STEP stpy
333 JI5     FOR o=0 TO 7
334 6S7       IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
335 fS9         IF pwy(o)=n AND pwx(o)=px THEN weg1=1
336 1n          IF pwy(o)=n AND pwx(o)=sx THEN weg2=1
337 217       ELSE
338 jX9         IF (pry(o)=n AND prx(o)=px) OR (pwy(o)=n AND pwx(o
            )=px) THEN weg1=1
339 Kv          IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o
            )=sx) THEN weg2=1
340 QJ7       END IF
341 8U5     NEXT o
342 7S3   NEXT n
343 57    IF weg1=1 AND weg2=1 THEN RETURN
344 D6    GOTO FZug2
345 b11 ShogunpruefX:
346 uA3   IF sx<px THEN stp=48
347 od    IF sx>px THEN stp=-48
348 2w    FOR n=sx+stp TO px-stp STEP stp
349 ZY5     FOR o=0 TO 7
350 Mi7       IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
351 UK9         IF pwx(o)=n AND pwy(o)=sy THEN RETURN
352 H07       ELSE
353 1W9         IF (prx(o)=n AND pry(o)=sy) OR (pwx(o)=n AND pwy(o
            )=sy) THEN RETURN
354 eX7       END IF
355 Mi5     NEXT o
356 Lg3   NEXT n
357 QJ    GOTO FZug2
358 r21 ShogunpruefY:
359 JA3   IF sy<py THEN stp=24
360 rh    IF sy>py THEN stp=-24
361 NJ    FOR n=sy+stp TO py-stp STEP stp
362 m15     FOR o=0 TO 7
363 Zv7       IF (sp=0 AND o=3) OR (sp=1 AND o=4) THEN
364 YS9         IF pwy(o)=n AND pwx(o)=sx THEN RETURN
365 UD7       ELSE
366 qZ9         IF (pry(o)=n AND prx(o)=sx) OR (pwy(o)=n AND pwx(o
            )=sx) THEN RETURN
367 rk7       END IF
368 Zv5     NEXT o
369 Yt3   NEXT n
370 n71 FZug2:
371 DM3   li=1
372 RF    IF spp<>sp THEN sb=sp:p$="Ihr Shogun wird bedroht!":GO
          SUB ShogunWirdBedroht:RETURN
373 uT    IF spp=sp THEN fz1$="Durch diesen Zug würde Ihr":fz2$="S
          hogun bedroht werden!":prx(m)=pxm:pry(m)=pym:zr(m)=zz:fz
          =0
374 oD    IF z<>-1 THEN pwx(z)=pxmem:pwy(z)=pymem:pm=0
375 E4    sx=sxx:sy=syy
376 vR0 FalscherZug:
377 KK3   IF mattpr=1 THEN MEnde=1:GOTO MattEnde
378 UU    IF li=1 AND ((sp=0 AND m=3) OR (sp=1 AND m=4)) THEN LINE
           (sx,sy)-(sx+47,sy+23),1+sp*3,b
379 dB    IF li=1 AND ((sp=0 AND m<>3) OR (sp=1 AND m<>4)) THE
          N LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23)
          ,1+sp*3,b
380 u5    IF li=1 THEN LINE (px,py)-(px+47,py+23),1+sp*3,b
381 kP    WINDOW OUTPUT 4+sp:CLS
382 Th    PRINT fz1$
383 1W    IF fz=1 THEN PRINT zar(m,zr(m));:fz=0
384 ap    PRINT fz2$
```

```
    WINDOW OUTPUT 2
    SOUND 150,5,255,1
    LINE (sx,sy)-(sx+47,sy+23),f(sx/48+sy/3),b
    LINE (px,py)-(px+47,py+23),f(px/48+py/3),b
    IF li=1 AND ((sp=0 AND m<>3) OR (sp=1 AND m<>4)) THE
    N LINE (prx(3+sp),pry(3+sp))-(prx(3+sp)+47,pry(3+sp)+23)
    ,f(prx(3+sp)/48+pry(3+sp)/3),b
    li=0
    GOTO CheckMouse2
ShogunWirdBedroht:
    LINE (sx,sy)-(sx+47,sy+23),1+sp*3,b:LINE (px,py)-(px+47,
    py+23),1+sp*3,b
    FOR x=230 TO 630 STEP 15
      SOUND x,.5,255,1:SOUND 830-x,.5,255,2
    NEXT x
    mattpr=1
    RETURN
FigurGeschlagen:
    FOR x=0 TO 7
      IF pwx(x)<>0 THEN Anzahl=Anzahl+1
    NEXT x
    IF sp=1 AND Anzahl<3 THEN sp=0:mattpr=0:GOTO VerlorenGe
    wonnen
    IF sp=0 AND Anzahl<3 THEN sp=1:mattpr=0:GOTO VerlorenGe
    wonnen
    Anzahl=0
    RETURN
WerteTauschen:
    FOR x=0 TO 7
      SWAP prx(x),pwx(x):SWAP pry(x),pwy(x):SWAP zr(x),zw(x)

      FOR y=1 TO 4
        SWAP zar(x,y),zaw(x,y)
      NEXT y
    NEXT x
    RETURN
Dunkel:
    FOR x=1 TO 0 STEP -.01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x
    NEXT x
    RETURN
Hell:
    FOR x=0 TO 1 STEP .01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x
    NEXT x
    RETURN
VerlorenGewonnen:
    WINDOW OUTPUT 4+sp:CLS
    PRINT na$(1+sp)","
    IF mattpr=1 THEN
      mattpr=0:spm=0:p$=""
      PRINT"Ihr Shogun ist matt!!!"
      PRINT"(Er würde beim nächsten Zug"
      PRINT"geschlagen werden.)"
      PRINT"Sie haben leider verloren."
    ELSE
      PRINT"Sie haben nur noch zwei"
      PRINT"Steine und somit verloren!"
      PRINT"Üben Sie noch ein wenig!"
    END IF
    WINDOW OUTPUT 5-sp:CLS
    PRINT na$(2-sp)","
    PRINT"Sie haben gewonnen!"
    PRINT"Herzlichen Glückwunsch!!!"
    WINDOW OUTPUT 3
    PRINT"Noch ein Spiel (J/N)?":i$="":p$=""
    WHILE i$<>"j" AND i$<>"n"
      i$=INKEY$
    WEND
    IF i$="j" THEN ns=1:GOTO NeuesSpiel
    IF i$="n" THEN SCREEN CLOSE 2:END
Daten0:
    DATA 4,1,4,1,2,3,2,3
    DATA 3,2,3,2,1,4,1,4
    DATA 2,3,2,3,4,1,4,1
    DATA 1,4,1,4,3,2,3,2
Daten90:
    DATA 3,4,3,4,1,2,1,2
    DATA 2,1,2,1,4,3,4,3
    DATA 1,2,1,2,3,4,3,4
    DATA 4,3,4,3,2,1,2,1
SUB Anleitung STATIC
    FOR x=1 TO 0 STEP -.01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x:PALETTE 2,x,x,x
    NEXT x
    CLS:COLOR 2,0
    PRINT TAB(33)"SPIELANLEITUNG":PRINT
    PRINT"Shogun ist ein Brettspiel, das zu zweit gespielt
    wird."
    PRINT
    PRINT"Jeder Spieler hat acht Steine, von denen der jewe
    ils gelb markierte der  SHOGUN"
    PRINT"(mit dem König im Schach vergleichbar) ist."
    PRINT"Die Zahlen auf den Steinen geben an, wieviele Fel
    der die Steine ziehen  dürfen."
    PRINT"Wenn ein Stein gesetzt wurde, ändert sich dessen
     Zahl  nach  einem  bestimmten"
    PRINT"System, das hier aber nicht näher erläutert wird,
     da das Spiel  sonst  an  Reiz"
    PRINT"verlieren würde. Auf dem Shogun können die  Zahle
    n 1 oder 2,  auf  den  anderen"
    PRINT"Steinen die Zahlen 1 bis 4 erscheinen."
    PRINT"Die Spieler ziehen  abwechselnd, Spieler 1 (rot)
    beginnt. Ein  Zug  kann  nicht"
    PRINT"verweigert werden."
    PRINT"Der Spieler klickt den Stein, den er ziehen möcht
    e, mit  der  linken  Maustaste"
    PRINT"an, bewegt den Mauspfeil dann auf das Feld, auf d
    as der  Stein  gesetzt  werden"
    PRINT"soll und drückt erneut die linke Maustaste."
    PRINT"Hat ein Spieler eine Figur angeklickt, die er doc
    h  nicht  setzen  möchte, kann"
    PRINT"er sie durch nochmaliges Anklicken wieder  loslas
    sen. Ein  bereits  vollständig"
    PRINT"ausgeführter Zug kann jedoch nicht zurückgenommen
     werden!"
    PRINT:PRINT
    PRINT TAB(33)"Weiter mit W"
    FOR x=0 TO 1 STEP .01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x
    NEXT x
    WHILE INKEY$<>"w"
    WEND
    FOR x=1 TO 0 STEP -.01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x
    NEXT x
    CLS:PRINT:PRINT
    PRINT"Die Steine dürfen nur waagerecht und senkrecht, n
    icht diagonal gezogen  werden."
    PRINT"Es dürfen weder gegnerische noch eigene Steine üb
    ersprungen werden.  Ein  Stein"
    PRINT"darf in einem Zug höchstens einmal seine Richtung
     um 90 ändern!"
    PRINT"Gegnerische Steine dürfen geschlagen werden."
    PRINT"Der Shogun darf nicht auf ein vom Gegner bedrohte
    s Feld gesetzt werden.Wenn der"
    PRINT"Shogun durch einen gegnerischen Stein bedroht wir
    d, muß er durch eine der  fol-"
    PRINT"genden Möglichkeiten in  Sicherheit  gebracht wer
    den:"
    PRINT"- er wird auf ein nicht bedrohtes Feld gesetzt"
    PRINT"- er schlägt den ihn bedrohenden Stein, wenn dies
    er nicht gedeckt ist"
    PRINT"- zwischen ihn und den ihn bedrohenden Stein wird
     ein anderer Stein gesetzt"
    PRINT
    PRINT"Verloren hat,"
    PRINT"- wer außer dem Shogun nur noch einen weiteren St
    ein besitzt oder"
    PRINT"- wessen Shogun so bedroht wird, daß dieser keine
     Möglichkeit mehr hat,sich von"
    PRINT"  der Bedrohung zu befreien."
    PRINT
    PRINT"Und nun viel Spaß beim Spiel!"
    PRINT
    PRINT TAB(33)"Weiter mit W"
    FOR x=0 TO 1 STEP .01
      PALETTE 0,x,x,x:PALETTE 1,x,x,x
    NEXT x
    WHILE INKEY$<>"w"
    WEND
END SUB
(C) 1987 M&T
```

Listing 1. (Schluß)



Grafik — Musik — DFÜ

Dieses im Rahmen der Commodore Sachbuchreihe erschienene Buch beschäftigt sich mit der Programmierung von Grafik, Musik und DFÜ auf dem Amiga. Vor dem Einstieg in diese Thematik erfährt der Leser etwas über die Geschichte des Amiga sowie Grundlagen zu den Themen Grafik und Amiga-DOS. Zur Verständlichkeit sowohl dieser als auch der weiteren Abschnitte des Buches tragen Informationskästen bei, in denen zusammengefaßt Grundwissen oder Anmerkungen zu einem bestimmten Thema vermittelt werden.

Der Autor David Myers führt in das gerade auf dem Amiga faszinierende Thema »Grafik« über die von der Programmiersprache Logo bekannte »Turtle Grafik« ein. Auch der »Animation« widmet er ein eigenes Kapitel. Leider bleibt D. Myers an der Oberfläche und geht nur kurz auf die Möglichkeiten ein, die Amiga-Basic mit seinen »Object«-Befehlen bietet.

Der zweite Schwerpunkt des Buches liegt bei der »Musik«. Unter der Überschrift »Sound« erlangt der Leser einiges an Hintergrundwissen zur Theorie der Tonerzeugung auf Computern allgemein, und speziell die Codierung von Tönen im Amiga. Dabei werden auch Begriffe wie »Amplitude« und »ADSR-Hüllkurve« erklärt. Der Leser erhält so einen Überblick über die wichtigsten Fachbegriffe dieses Gebietes. Innerhalb dieses Teiles behandelt David Myers auch das Thema Sprache und die Verwendung von Phonemen.

Ein eigenes Kapitel bildet die Abhandlung über »Synthetische Musik«. Hier stellt der Autor zum Beispiel Tonwellenformen vor. Nebenbei lernt der Leser, wie sich etwa die Titelmusik von »Star Wars« auf dem Amiga programmieren läßt. Wer sich also dem Thema Musik auf dem Amiga widmen will, findet hier eine Vielzahl nützlicher Informationen.

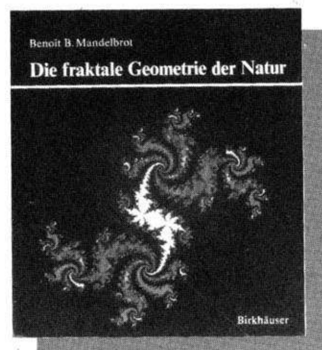
Der letzte Schwerpunkt des Buches befaßt sich mit der Datenfernübertragung (DFÜ). Neben obligatorischem Grundwissen zeigt der Autor dort, wie ein Terminalprogramm in Basic erstellt werden kann. Ergänzend fügt er eine Liste mit während der DFÜ eventuell auftretenden Fehlern und deren Behebung hinzu. Eine Beschreibung wichtiger Mailboxen und Informationsdienste rundet das Kapitel ab.

Wer die Programmierung der drei Gebiete »Grafik — Musik — DFÜ« mit Amiga-Basic lernen will, findet in diesem Buch sicher geeignete Unterstützung, zumal es zu den einzelnen Themen Programmbeispiele enthält. Diese liegen dem Buch auf einer Diskette bei. (Ingolf Krüger/rs)

Amiga Grafik — Musik — DFÜ; David Myers; Markt & Technik Verlag AG; 225 Seiten; ISBN 3-89090-579-x, Preis 59 Mark

Die fraktale Geometrie der Natur

Wer kennt sie nicht, die »Apfelmännchen«. Unter diesem Namen wurden die selbstähnlichen Abbildungen (eine Untermenge der »Fraktale«) einem größeren Publikum bekannt. Gerade auf dem Amiga existieren eine Vielzahl von Programmen, die diese teilweise faszinierenden Grafiken erzeugen. Entdeckt wurden Sie von Benoit Mandelbrot, seines Zeichens Professor für Mathematik an der Harvard Universität. Unter dem Titel »Die fraktale Geometrie der Natur« beschreibt er die den Fraktalen zugrunde liegende Theorie und definiert eine Vielzahl von Begriffen. Sein Ziel ist die Verknüpfung natürlicher Strukturen mit seinem Prinzip der Selbstähnlichkeit. Der Hintergrund der Fraktale ist jedoch (oft recht aufwendige) Mathematik. Und so behandelt dieses Buch auch nicht die Umsetzung der verschiedenen Fraktale auf den Computer, sondern schildert deren mathematischen Hintergrund. Es enthält eine Vielzahl von Abbildungen, die teilweise so fantastisch realistisch sind, daß sie dem Leser eine nähere Beschäftigung mit dem Begriff Fraktal nahezu aufzwingen. Die Übersetzung aus dem Englischen ist gut und flüssig zu lesen. Nach einer kurzen Einleitung und der Vorstellung erster Fraktale und deren Problemstellungen führt Benoit. B. Mandelbrot den Leser zu Themen wie »Galaxien und Wirbel«, »skaleninvariante Fraktale« und »nichtskaleninvariante Fraktale«. Ausführliche Quellen-, Literatur- und Stichwortverzeichnisse runden das umfangreiche Werk ab.



Dieses Buch ist nicht dazu geeignet, auf die schnelle den Schlüssel zur Umsetzung von Apfelmännchen und anderen fraktalen Strukturen auf den heimischen Computer zu liefern. Wer sich jedoch mit dem nötigen mathematischen Grundwissen und (wissenschaftlichem) Interesse an das Buch heranwagt, wird schon wegen der zahlreichen Abbildungen und der vielfältigen Erscheinungsformen der Fraktale das Buch schätzen. (Ingolf Krüger/rs)

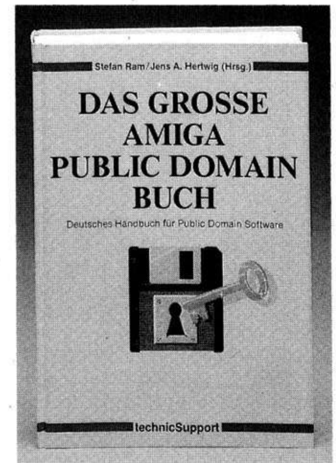
Die fraktale Geometrie der Natur; Benoit B. Mandelbrot; Birkhäuser Verlag; 491 Seiten; ISBN 3-7643-1771-X; Preis 118 Mark

Das große Amiga-Public Domain-Buch

Dieses Buch ist allen Amiga-Benutzern gewidmet, die sich näher mit dem breiten Spektrum der Public Domain-Programme beschäftigen möchten. Es gibt Hilfestellung bei vielen Fragen, die zu diesem Thema auftreten.

In ersten Teil des Buchs wird durch eine Begriffserläuterung erklärt, was es mit den Bezeichnungen »Public Domain« und »Shareware« auf sich hat und welche Regeln beim Umgang mit Public Domain-Disketten zu beachten sind. Des weiteren folgt eine kurze Unterweisung im Umgang mit dem Amiga. Es werden aber nur die nötigsten Handgriffe beschrieben.

Der zweite Teil, der mit etwa 200 Seiten am umfangreichsten ausgefallen ist, enthält sehr ausführliche deutsche



Beschreibungen, Anleitungen und Quellenangaben zu 43 PD-Programmen. Dabei gliedert sich jede Programmbesprechung in die Erklärung des Programmaufrufs, den Zweck der jeweiligen Routine, die Herkunftsbestimmung und die eigentliche Bedienung des Programms. Dabei ist die Erklärung sehr ausführlich geworden. Einige Programme wie beispielsweise das Rollenspiel »Hack« oder das DFÜ-Programm »Kermit« werden auf jeweils über zehn Buchseiten beschrieben. Der in Englisch nicht so bewanderte Anwender erhält somit die Chance, komplexere PD-Programme besser auszunutzen, da ihm genauere Informationen zum jeweiligen Programm offeriert werden.

In dritten Teil des ersten Bandes, der mit »Listen« umschrieben ist, findet der Käufer ein Inhaltsverzeichnis von drei PD-Serien: Fish 1 bis 127, Panorama 1 bis 17b und FAUG 1 bis 39. Das Verzeichnis, in dem zu jedem Programm ein Kurzkommentar enthalten ist, findet sich in dreifacher Ausführung: einmal ist es nach Themen sortiert, ein anderes Mal alphabetisch geordnet und in der dritten Ausführung nach Diskettenserien gegliedert. Damit findet jeder Leser schnell das Programm und die Diskettennummer der für ihn interessanten Routine.

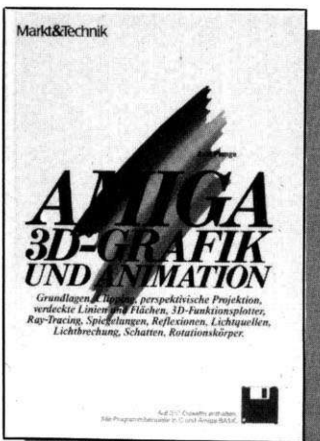
Weitere Bücher dieser Serie sind bereits in Arbeit, so daß sich jeder Interessent eine sehr sinnvolle und hilfreiche Bibliothek in sein Regal stellen kann. Der zweite Band ist seit Herbst 1988 auf dem deutschen Markt, eine Besprechung folgt. (Jens Krumbeck/rs)

Das große Amiga Public Domain Buch — Band 1, Stefan Ram/Jens A. Hertwig, technicSupport-Verlag, 350 Seiten, ISBN 3-926847-01-8, Preis 49 Mark

Amiga 3D-Grafik und Animation

Um anspruchsvolle 3D-Grafiken zu erstellen, ist sowohl mathematisches Grundwissen als auch eine Kenntnis der Grundlagen zur Erzeugung von Grafiken mit dem Computer notwendig. Beides vermittelt auf anschauliche Weise das Buch »Amiga 3D-Grafik und Animation« von Axel Plenge. Nach einer kurzen Einführung stellt der Autor bereits die wichtigsten Grundlagen zum Aufbau einer Farbgrafik auf dem Amiga vor, ohne sich jedoch in technischen Fachwörtern und ausschweifenden Schilderungen zu ergehen. Im dritten Kapitel werden sodann die Grundlagen für die zweidimensionale Grafik gelegt. In diesem Kapitel geht der Autor auch in puncto Mathematik »in medias res«. Doch alle mathematischen Grundlagen werden auf eine sehr anschauliche und dadurch verständliche Weise dargebracht. Im Anhang findet sich zusätzlich ein Abschnitt, der die für die Computergrafik wichtige Mathematik (Winkelfunktionen wie Sinus, Cosinus und Matrizenrechnung) wiederholt. Selbst Mathematikmuffel werden kaum mit Verständnisproblemen zu kämpfen haben.

Neben dem mathematischen Grundwissen erfährt der Leser bereits in Kapitel 3 von Themen wie Vergrößerung/Verkleinerung, Spiegelung, Drehung, Translation und Clipping von grafischen Darstellungen. Kapitel 4 bringt dann den Einstieg in die dreidimensionale Bildschirmgrafik. Der Autor leitet nach einem Ausflug in die »Welt« der Koordinatensysteme und mathematischen Grundlagen von Transformationen zweidimensionaler Grafik (Spiegelung, Drehung...) über auf den dreidimensionalen Raum.

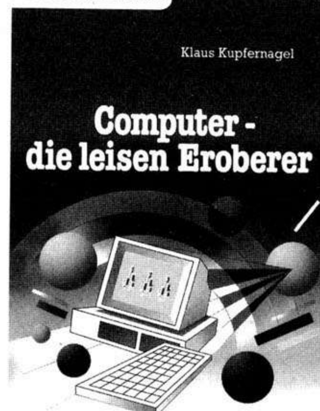


Kapitel 5 widmet sich dem Problem der verdeckten Linien und Flächen. Daran schließt sich ein Kapitel an, das vollständig dem Thema Ray-Tracing gewidmet ist. Den Abschluß bildet die Behandlung von Rotationskörpern. Das durchweg gelungene Werk wird durch den bereits erwähnten Anhang, der auch eine Schilderung der verwendeten Library-Routinen enthält, zahlreiche Literaturhinweise und ein ausführliches Stichwortverzeichnis abgerundet.

(Ingolf Krüger/rs)

Amiga 3D-Grafik und Animation; Axel Plenge; Markt & Technik Verlag AG; 376 Seiten; ISBN 3-89090-526-9; Preis 69 Mark

Computer — die leisen Eroberer



Ob im Büro oder Zuhause — der Computer hat unwiderruflich seinen Platz in unserer Gesellschaft eingenommen. Doch Angst und Mißtrauen begleiten den lautlosen Einzug in unser Leben.

Klaus Kupfernagel will dem mit seinem Buch entgegenwirken. »Der Computer«, schreibt er, »wird unsere Gesellschaft verändern, aber das Wissen dieser Gesellschaft über die Computer ist gering.« Kupfernagel erläutert gezielt Funktion, Möglichkeiten, Auswirkungen und Grenzen des Schreckgespenstes »Denkmaschine«. Er vermittelt somit das nötige Wissen für die erfolgreiche Auseinandersetzung mit den Konsequenzen und Gefahren der Computer.

Das Buch ist durchaus kein Lehrbuch. Allein die lockere, witzig-spritzige Art und Weise, in der das Buch geschrieben ist, macht es lesenswert. Viele anschauliche Beispiele bringen auch Einsteigern die Bits und Bytes, RAMs und ROMs, Floppys, Zentraleinheiten und andere Feinheiten nahe.

(Jan-Dirk Mittmann/rs)

Computer — die leisen Eroberer, Klaus Kupfernagel, Markt & Technik Verlag, 270 Seiten, ISBN 3-89090-179-4, 14,80 Mark

Grafik auf dem Amiga

»Grafik auf dem Amiga« ist in fünf Teile gegliedert, die unabhängig voneinander gelesen werden können. Der Einsteiger kann sich zunächst den Grundlagen der Grafik zuwenden. Er erfährt hier etwas über grundsätzliche Techniken der Bilderzeugung auf einem Monitor/Fernseher und gelangt unversehens in die mathematisch/geometrische Ableitung von Algorithmen zur Erzeugung von Linien, Kreisen und dreidimensionalen Objekten.

Die Hardwarespezialisten finden im folgenden Kapitel eine Beschreibung der Amiga-Hardware.

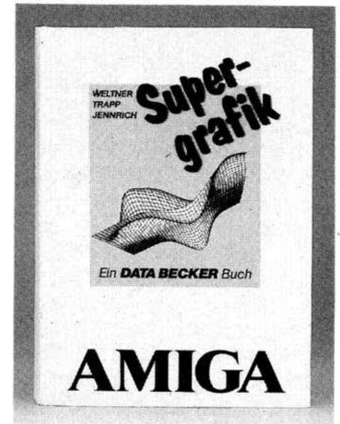
Der dritte Teil beschäftigt sich mit den Möglichkeiten des Amiga-Basic, die Grafikhardware zu beeinflussen. Da Grundkenntnisse der Programmiersprache vorausgesetzt werden, hält sich der Autor auch nicht lange mit den grundsätzlichen Techniken auf, sondern gelangt recht schnell in das interessante Gebiet der Computeranimation, welches dann um so ausführlicher behandelt wird.

Ein weiterer Teil führt in die grundsätzliche Arbeitsweise und Struktur des Betriebssystems beziehungsweise der von der Grafikdarstellung betroffenen Systembibliotheken ein. Wer mehr über die Funktionsweise der Amiga-Grafik wissen möchte, kommt um dieses Buch nicht herum.

(Joschy Polierer/rs)

Grafik auf dem Amiga, Manfred Kohlen, Verlag Markt & Technik AG, 340 Seiten, ISBN 3-89090-236-7, Preis: 49 Mark





Supergrafik

Unbestritten lassen sich mit dem Amiga fantastische Grafiken auf den Bildschirm zaubern. Wer solche Kunstwerke allerdings selbst programmieren möchte, der benötigt fundierte Kenntnisse über die Systemgrafikroutinen des Amiga. Dieses Wissen vermittelt das vorliegende Buch.

Je nach Vorkenntnissen bietet die Aufteilung des Werkes jeweils drei Einstiegspunkte für Anfägner, Fortgeschrittene und angehende Profis. Im ersten Teil wird der Anfänger mit den elementaren Grafikbefehlen des Amiga-Basic bekannt gemacht. Er lernt, wie sich Grafiken mit den Mitteln, die Basic zur Verfügung stellt, effektiv erstellen lassen. Im zweiten Teil wird dem fortgeschrittenen Basic-Programmierer gezeigt, wie er die Grafikroutinen des Betriebssystems in Basic-Programmen nutzen kann. Mit deren Hilfe ist es dann möglich, von Basic aus den 64farbigen Halfbrite- und den 4096-farbigen H.A.M.-Modus einzuschalten, den Coprozessor Copper zu programmieren und Grafik auf dem Drucker im Multitasking-Betrieb auszugeben. Der dritte Teil wendet sich an die C-Programmierer und zeigt, wie man Screens und Fenster in C erstellt, wie Sprites, BOBs und VSprites programmiert und dann mit Hilfe des vollautomatischen Animationssystems des Amiga in Bewegung versetzt werden.

»Supergrafik« enthält alle wichtigen Informationen, die ein Programmierer benötigt, wenn er in die Welt der Amiga-Grafik einsteigen will. Es kann deshalb jedem, der sich für die Grafikprogrammierung mit dem Amiga interessiert, empfohlen werden.

(Christian Schneider/rs)

Supergrafik, Weltner/Trapp/Jennrich, Data Becker, 690 Seiten, ISBN 3-89011-254-4, Preis: 59 Mark

Fortsetzung von Seite 8

Sprite-Editor

Das Programm erkennt, ob es sich um ein »AOE«-Objekt (Amiga-Object-Editor-Objekt), oder um einen DPaint-Brush handelt. DPaint-Brushes müssen in der Größe von 32 x 32 Punkten mit 16 Farben vorliegen. Wollen Sie selbsterstellte Sprites auf Diskette ablegen, so wählen Sie »SPEICHERN«. Hier sind die Nummern des ersten und des letzten zu speichernden Objekts angegeben. Es können also beliebig viele aufeinanderfolgende Sprites in einem File gespeichert werden. Wollen Sie nur ein Sprite speichern, geben Sie einfach zwei gleiche Werte an. Nach Eingabe des Dateinamens werden die Objekte im »AOE«-Format gespeichert. Den Aufbau eines »AEO«-Files sehen Sie in Tabelle 1.

Zahlreiche Icons erleichtern die Arbeit

Icon 2 (CLR): Wenn Sie dieses Icon anwählen, wird das aktuelle Sprite gelöscht.
Icon 3 (zwei waagerechte Pfeile): Horizontales Spiegeln des aktuellen Sprites.
Icon 4 (zwei senkrechte Pfeile): Vertikales Spiegeln des aktuellen Sprites.
Icon 5 (geknickter Pfeil): Drehen des aktuellen Sprites um 90 Grad.
Icon 6 (zwei sich überlappende Quadrate): Kopieren von Sprites. Neben der Quell- und Zielnummer des Objekts kann beim Kopieren von mehreren aufeinanderfolgenden Sprites auch die gewünschte Anzahl eingegeben werden.

Kopieren und Animieren von Sprites

Icon 7 (»i«): Info zum Programm.
Icon 8 (Bildprojektor-Symbol): Animation von mehreren Sprites. Geben Sie die Nummern des ersten und letzten Sprites der Animationsfolge ein und klicken Sie »FERTIG« an. Daraufhin beginnt die Animation, deren Geschwindigkeit mit der Maus verändert werden kann. Die Zahl zwischen den Pfeilen gibt die Geschwindigkeit in $\frac{1}{50}$-Sekunden an.

Haben Sie Ihre Arbeit beendet und wollen den Editor verlassen, bewegen Sie die Maus in die rechte obere Bildschirmecke und drücken den rechten Mausknopf.

Nun steht dem munteren Erstellen von Sprites und Animationssequenzen nichts mehr im Wege. Grenzen setzt nur die eigene Kreativität — und von der besitzen Amiga-Besitzer ja eine ganze Menge.

(Michael Bertsch/C. Ewald/M.Jobst/rs)

So geben Sie das Listing ein

Der Objekt-Editor ist in Maschinensprache (Assembler) geschrieben. Um jedem Amiga-Besitzer die Eingabe des Listings zu ermöglichen, haben wir das Programm für Sie umgewandelt. Die abgedruckte Version ist mit Amiga-Basic einzugeben. Diesen Interpreter finden Sie auf der Extras-Diskette, die bei der Auslieferung jedem Amiga beiliegt. Das Programm erzeugt beim Start das lauffähige Programm mit Namen »AOE_V1.2« auf der eingelegten Diskette. Bitte geben Sie das Programm unbedingt mit Checkie 42 (Seite 159) ein. Die Hinweise zum Checksummer sollten Sie unbedingt beachten.

Programmname:	AOE_Gen
Computer:	Amiga 500,1000,2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

```
tIO REM Generiert Programm AOE_V1.2
ag  CLS
gO  OPEN "AOE_V1.2" FOR OUTPUT AS 1
BS  READ anz
oa  FOR i=1 TO anz
3n1  READ h$
yB2   wert1=ASC(LEFT$(h$,1))
FD    IF wert1>64 THEN wert1=wert1-55 ELSE wert1=wert1-48
FI    wert1=wert1*16
7c    wert2=ASC(RIGHT$(h$,1))
ad    IF wert2>64 THEN wert2=wert2-55 ELSE wert2=wert2-48
Pi    wert=wert1+wert2
9G    PRINT #1,CHR$(wert);
```

```
JOO NEXT
3n  CLOSE 1
Ov  END
R4  DATA 19072
ie  DATA 00,00,03,F3,00,00,00,00,00,00
aj  DATA 00,02,00,00,00,00,00,00,00,01
UF  DATA 00,00,0F,1A,00,00,01,00,00,00
gb  DATA 03,E9,00,00,0F,1A,4E,F9,00,00
72  DATA 2B,94,2C,78,00,04,43,F9,00,00
ut  DATA 3B,30,42,80,4E,AE,FD,D8,23,C0
Sy  DATA 00,00,3A,3E,43,F9,00,00,3B,41
az  DATA 42,80,4E,AE,FD,D8,23,C0,00,00
6d  DATA 3A,42,43,F9,00,00,3B,4D,42,80
6p  DATA 4E,AE,FD,D8,23,C0,00,00,3A,46
N4  DATA 4E,75,2C,78,00,04,22,79,00,00
D7  DATA 3A,3E,4E,AE,FE,62,22,79,00,00
Tc  DATA 3A,42,4E,AE,FE,62,22,79,00,00
bf  DATA 3A,46,4E,AE,FE,62,4E,75,2C,79
KH  DATA 00,00,3A,46,4E,AE,FF,B2,41,F9
gd  DATA 00,00,2D,46,4E,AE,FF,3A,23,C0
9F  DATA 00,00,2D,84,23,C0,00,00,2D,B4
zt  DATA 23,C0,00,00,2D,E4,41,F9,00,00
PL  DATA 2D,66,4E,AE,FF,34,23,C0,00,00
fb  DATA 3A,52,20,40,4E,AE,FE,D4,23,C0
cK  DATA 00,00,3A,62,20,79,00,00,3A,52
W3  DATA 23,E8,00,32,00,00,3A,66,23,E8
BS  DATA 00,32,00,00,3A,6E,20,68,00,32
ge  DATA 23,E8,00,04,00,00,3A,5A,2C,79
4I  DATA 00,00,3A,3E,20,79,00,00,3A,62
v1  DATA 43,F9,00,00,32,6E,30,3C,00,20
2u  DATA 4E,AE,FF,40,2C,79,00,00,3A,46
oE  DATA 4E,AE,FE,80,4E,75,2C,79,00,00
kU  DATA 3A,46,20,79,00,00,3A,52,4E,AE
QW  DATA FF,B8,20,79,00,00,2D,84,4E,AE
63  DATA FF,BE,4E,AE,FF,2E,4E,75,2C,78
2p  DATA 00,04,93,C9,4E,AE,FE,DA,28,40
yZ  DATA 4A,AC,00,AC,66,00,00,18,41,EC
pR  DATA 00,5C,4E,AE,FE,80,41,EC,00,5C
hO  DATA 4E,AE,FE,8C,23,C0,00,00,3A,82
aC  DATA 4E,75,4A,B9,00,00,3A,82,67,00
A1  DATA 00,0C,22,79,00,00,3A,82,4E,AE
HD  DATA FE,86,42,80,4E,75,2C,79,00,00
bd  DATA 3A,3E,43,F9,00,00,3A,98,4E,AE
71  DATA FF,3A,23,FC,00,00,3B,08,00,00
C1  DATA 3A,9C,22,79,00,00,3A,66,30,3C
fz  DATA 00,10,4E,AE,FE,AA,22,79,00,00
8j  DATA 3A,66,42,80,42,81,34,3C,00,E0
WD  DATA 36,3C,00,E0,4E,AE,FE,CE,22,79
Dd  DATA 00,00,3A,66,30,3C,00,02,4E,AE
UQ  DATA FE,AA,22,79,00,00,3A,66,42,80
RH  DATA 4E,AE,FE,A4,22,79,00,00,3A,66
jK  DATA 10,3C,00,00,4E,AE,FE,9E,42,86
H1  DATA 22,79,00,00,3A,66,33,46,00,24
nV  DATA 33,7C,00,00,00,26,30,06,32,3C
xL  DATA 00,E0,4E,AE,FF,0A,22,79,00,00
DZ  DATA 3A,66,33,7C,00,00,00,24,33,46
HO  DATA 00,26,30,3C,00,E0,32,06,4E,AE
AZ  DATA FF,0A,DC,7C,00,07,BC,7C,00,E7
9q  DATA 66,C2,22,79,00,00,3A,66,30,3C
CF  DATA 00,04,4E,AE,FE,AA,22,79,00,00
1U  DATA 3A,66,33,7C,00,70,00,24,33,7C
KC  DATA 00,01,00,26,30,3C,00,70,32,3C
85  DATA 00,DF,4E,AE,FF,0A,22,79,00,00
dG  DATA 3A,66,33,7C,00,01,00,24,33,7C
tV  DATA 00,70,00,26,30,3C,00,DF,32,3C
OA  DATA 00,70,4E,AE,FF,0A,22,79,00,00
BR  DATA 3A,66,30,3C,00,01,4E,AE,FE,AA
08  DATA 22,79,00,00,3A,66,30,3C,00,E1
E1  DATA 42,81,34,3C,01,3F,36,3C,00,FF
UT  DATA 4E,AE,FE,CE,22,79,00,00,3A,66
8H  DATA 30,3C,00,00,32,3C,00,E1,34,3C
d1  DATA 00,E1,36,3C,00,FF,4E,AE,FE,CE
oc  DATA 22,79,00,00,3A,66,30,3C,00,02
EH  DATA 4E,AE,FE,AA,22,79,00,00,3A,66
3i  DATA 30,3C,00,01,4E,AE,FE,A4,30,3C
nL  DATA 00,02,38,3C,00,FF,3A,3C,00,07
H1  DATA 3C,3C,01,20,3E,3C,00,28,4E,B9
VC  DATA 00,00,04,88,30,3C,00,04,38,3C
QS  DATA 00,FE,3A,3C,00,06,3C,3C,01,21
R5  DATA 3E,3C,00,29,4E,B9,00,00,04,88
qd  DATA 22,79,00,00,3A,66,30,3C,00,10
MP  DATA 4E,AE,FE,AA,22,79,00,00,3A,66
pl  DATA 30,3C,01,00,32,3C,00,08,34,3C
v5  DATA 01,1F,36,3C,00,27,4E,AE,FE,CE
fW  DATA 4B,F9,00,00,32,AE,3C,3C,00,10
tP  DATA 30,06,22,79,00,00,3A,66,4E,AE
jx  DATA FE,AA,30,1D,32,1D,34,1D,36,1D
Xn  DATA 22,79,00,00,3A,66,4E,AE,FE,CE
II  DATA DC,7C,00,01,BC,7C,00,20,66,D8
kE  DATA 30,3C,00,02,38,3C,00,0F,3A,3C
1U  DATA 00,E7,3C,3C,00,1B,3E,3C,00,F9
4E  DATA 4E,B9,00,00,04,88,30,3C,00,04
Zk  DATA 38,3C,00,0E,3A,3C,00,E6,3C,3C
Oi  DATA 00,1C,3E,3C,00,FA,4E,B9,00,00
Uh  DATA 04,88,2C,79,00,00,3A,46,20,79
WR  DATA 00,00,3A,66,43,F9,00,00,34,36
mH  DATA 30,3C,01,09,32,3C,00,50,4E,AE
Tf  DATA FF,8E,43,F9,00,00,34,FE,20,79
D2  DATA 00,00,3A,66,30,3C,01,00,32,3C
Ue  DATA 00,66,4E,AE,FF,8E,43,F9,00,00
an  DATA 35,9E,20,79,00,00,3A,66,30,3C
fH  DATA 00,F8,32,3C,00,BC,4E,AE,FF,8E
NL  DATA 43,F9,00,00,35,C6,20,79,00,00
pZ  DATA 3A,66,30,3C,00,FB,32,3C,00,E2
Di  DATA 4E,AE,FF,8E,4B,F9,00,00,3B,5F
Rn  DATA 3A,3C,00,0A,3C,3C,00,E8,3E,3C
QW  DATA 00,B4,4E,B9,00,00,04,18,4B,F9
Ts  DATA 00,00,3B,74,3A,3C,00,03,3C,3C
EI  DATA 01,03,3E,3C,00,C2,4E,B9,00,00
Eh  DATA 04,18,4B,F9,00,00,3B,69,3A,3C
yj  DATA 00,05,3C,3C,00,FB,3E,3C,00,DE
21  DATA 4E,B9,00,00,04,18,4E,B9,00,00
```

```
126 R3  DATA 05,C2,4E,75,2C,79,00,00,3A,3E
127 zi  DATA 22,79,00,00,3A,6E,33,46,00,24
128 ps  DATA 33,47,00,26,30,3C,00,01,4E,AE
129 G3  DATA FE,9E,22,79,00,00,3A,6E,42,80
130 8Q  DATA 4E,AE,FE,AA,22,79,00,00,3A,6E
131 mS  DATA 20,4D,30,05,4E,AE,FF,C4,22,79
132 GK  DATA 00,00,3A,6E,DC,7C,00,01,DE,7C
133 Db  DATA 00,01,33,46,00,24,33,47,00,26
134 b4  DATA 42,80,4E,AE,FE,9E,22,79,00,00
135 cw  DATA 3A,6E,30,3C,00,03,4E,AE,FE,AA
136 up  DATA 22,79,00,00,3A,6E,20,4D,30,05
137 YL  DATA 4E,AE,FF,C4,4E,75,2C,79,00,00
138 07  DATA 3A,3E,22,79,00,00,3A,6E,4E,AE
139 gw  DATA FE,AA,22,79,00,00,3A,6E,33,44
140 Ra  DATA 00,24,33,45,00,26,30,06,32,05
141 j8  DATA 4E,AE,FF,0A,22,79,00,00,3A,6E
142 Qv  DATA 30,06,32,07,4E,AE,FF,0A,22,79
143 wJ  DATA 00,00,3A,6E,30,04,32,07,4E,AE
144 5y  DATA FF,0A,22,79,00,00,3A,6E,30,04
145 S1  DATA 32,05,4E,AE,FF,0A,4E,75,42,87
146 yx  DATA 41,F9,00,00,32,AE,30,18,32,18
147 3p  DATA 34,18,36,18,B0,45,62,00,00,1A
148 i3  DATA B2,46,62,00,00,14,B4,45,65,00
149 no  DATA 00,0E,B6,46,65,00,00,08,4E,F9
150 XR  DATA 00,00,05,20,DE,7C,00,01,BE,7C
151 ds  DATA 00,10,66,D0,08,39,00,06,00,BF
152 HY  DATA E0,01,67,F6,60,00,26,FE,23,F9
153 C9  DATA 00,00,3A,66,00,00,3A,6E,33,C7
154 PP  DATA 00,00,3A,74,3E,39,00,00,3A,72
155 1v  DATA CE,FC,00,08,28,47,D9,FC,00,00
156 N8  DATA 32,AE,61,00,00,62,61,00,00,68
157 00  DATA 30,3C,00,01,61,00,FF,38,61,00
158 yZ  DATA 00,5C,30,3C,00,01,61,00,FF,2C
159 rd  DATA 42,87,3E,39,00,00,3A,74,33,C7
160 ug  DATA 00,00,3A,72,CE,FC,00,08,28,47
161 SN  DATA D9,FC,00,00,32,AE,61,00,00,2C
162 AR  DATA 61,00,00,32,30,3C,00,02,61,00
163 Fn  DATA FF,02,61,00,00,26,30,3C,00,04
164 b7  DATA 61,00,FE,F6,61,00,00,2C,08,39
165 IQ  DATA 00,06,00,BF,E0,01,67,F6,60,00
166 Bq  DATA 26,78,38,1C,3A,1C,3C,1C,3E,1C
167 CH  DATA 4E,75,98,7C,00,01,9A,7C,00,01
168 pz  DATA DC,7C,00,01,DE,7C,00,01,4E,75
169 TU  DATA 42,80,30,39,00,00,3A,72,C0,FC
170 HY  DATA 00,02,22,40,D3,FC,00,00,32,8E
171 8N  DATA 30,11,3E,00,61,00,00,48,13,C0
172 98  DATA 00,00,3B,72,30,07,E8,48,61,00
173 wu  DATA 00,3A,13,C0,00,00,3B,70,30,07
174 Dw  DATA E0,48,61,00,00,2C,13,C0,00,00
175 HS  DATA 3B,6E,4B,F9,00,00,3B,6E,3A,3C
176 Vv  DATA 00,06,3C,3C,00,FB,3E,3C,00,F2
177 gp  DATA 23,F9,00,00,3A,66,00,00,3A,6E
178 vb  DATA 4E,B9,00,00,04,18,4E,75,C0,7C
179 x3  DATA 00,0F,D0,7C,00,30,B0,7C,00,3A
180 L0  DATA 65,00,00,06,D0,7C,00,07,4E,75
181 Ao  DATA 2C,79,00,00,3A,3E,8A,FC,00,07
182 5q  DATA 8C,FC,00,07,CA,BC,00,00,FF,FF
183 yR  DATA CC,BC,00,00,FF,FF,20,05,22,06
184 or  DATA C0,FC,00,07,C2,FC,00,07,D0,7C
185 1q  DATA 00,01,D2,7C,00,01,34,00,36,01
186 D4  DATA D4,7C,00,05,D6,7C,00,05,48,E7
187 jZ  DATA F0,00,30,39,00,00,3A,72,D0,7C
188 Jb  DATA 00,10,22,79,00,00,3A,66,4E,AE
189 cN  DATA FE,AA,22,79,00,00,3A,66,4C,DF
190 4G  DATA 00,0F,4E,AE,FE,CE,43,F9,00,00
191 Vw  DATA 3A,98,30,39,00,00,3A,72,D0,7C
192 nK  DATA 00,10,4E,AE,FE,AA,43,F9,00,00
193 Ws  DATA 3A,98,30,05,32,06,4E,AE,FE,BC
194 Bc  DATA 22,79,00,00,3A,66,30,05,32,06
195 EJ  DATA D0,7C,01,00,D2,7C,00,08,4E,AE
196 eX  DATA FE,BC,4E,75,42,87,41,F9,00,00
197 Hd  DATA 33,2E,30,18,32,18,34,18,36,18
198 JP  DATA B0,45,62,00,00,1A,B2,46,62,00
199 Bz  DATA 00,14,B4,45,65,00,00,0E,B6,46
200 gY  DATA 65,00,00,08,4E,F9,00,00,07,1A
201 qr  DATA DE,7C,00,01,BE,7C,00,16,66,D0
202 4C  DATA 08,39,00,06,00,BF,E0,01,67,F6
203 qn  DATA 60,00,25,04,CE,FC,00,04,20,47
204 Jc  DATA D1,FC,00,00,33,DE,22,50,4E,D1
205 7F  DATA 08,39,00,06,00,BF,E0,01,67,F6
206 9I  DATA 23,F9,00,00,3A,66,00,00,3A,6E
207 hk  DATA 60,00,24,DC,0C,79,00,01,00,00
208 2D  DATA 3A,7C,67,00,01,84,20,79,00,00
209 DM  DATA 3B,10,61,00,00,26,20,79,00,00
210 DI  DATA 3B,14,61,00,00,1C,20,79,00,00
211 nL  DATA 3B,18,61,00,00,12,20,79,00,00
212 eY  DATA 3B,1C,61,00,00,08,61,00,20,2E
213 Vd  DATA 60,AE,2E,28,00,7C,24,48,22,48
214 7v  DATA D3,FC,00,00,00,80,D1,FC,00,00
215 hI  DATA 00,7C,20,3C,00,00,00,1E,23,20
216 5d  DATA 51,C8,FF,FC,24,87,4E,75,0C,79
217 bR  DATA 00,01,00,00,3A,7C,67,00,01,78
218 3Q  DATA 20,79,00,00,3B,10,61,00,00,28
219 Fs  DATA 20,79,00,00,3B,14,61,00,00,1E
220 5V  DATA 20,79,00,00,3B,18,61,00,00,14
221 cP  DATA 20,79,00,00,3B,1C,61,00,00,0A
222 hU  DATA 61,00,1F,D0,60,00,FF,50,2E,10
223 93  DATA 24,48,22,48,D1,FC,00,00,00,04
224 2z  DATA 20,3C,00,00,00,1E,22,D8,51,C8
225 Ox  DATA FF,FC,25,47,00,7C,4E,75,0C,79
226 j1  DATA 00,01,00,00,3A,7C,67,00,01,6E
227 CZ  DATA 20,79,00,00,3B,10,61,00,00,28
228 01  DATA 20,79,00,00,3B,14,61,00,00,1E
229 Ee  DATA 20,79,00,00,3B,18,61,00,00,14
230 1Y  DATA 20,79,00,00,3B,1C,61,00,00,0A
231 s2  DATA 61,00,1F,76,60,00,FE,F6,20,3C
232 Hb  DATA 00,00,00,1F,22,10,E2,99,20,C1
233 Pc  DATA 51,C8,FF,F8,4E,75,0C,79,00,01
234 9z  DATA 00,00,3A,7C,67,00,01,62,20,79
235 ea  DATA 00,00,3B,10,61,00,00,28,20,79
236 am  DATA 00,00,3B,14,61,00,00,1E,20,79
237 dc  DATA 00,00,3B,18,61,00,00,14,20,79
238 2t  DATA 00,00,3B,1C,61,00,00,0A,61,00
239 1U  DATA 1F,28,60,00,FE,A8,20,3C,00,00
240 GH  DATA 00,1F,22,10,E3,99,20,C1,51,C8
241 rH  DATA FF,F8,4E,75,0C,79,00,01,00,00
242 x4  DATA 3A,7C,67,00,01,5A,2C,79,00,00
243 0Q  DATA 3A,46,20,79,00,00,3A,66,43,F9
244 fH  DATA 00,00,34,9A,20,3C,00,00,01,09
245 wt  DATA 22,3C,00,00,00,50,4E,AE,FF,8E
246 tN  DATA 33,FC,00,01,00,00,3A,7C,60,00
247 fJ  DATA FE,5C,20,79,00,00,3B,10,61,00
248 u0  DATA 00,28,20,79,00,00,3B,14,61,00
249 Mf  DATA 00,1E,20,79,00,00,3B,18,61,00
250 R1  DATA 00,14,20,79,00,00,3B,1C,61,00
251 En  DATA 00,0A,61,00,1E,AC,60,00,FE,2C
252 ie  DATA 24,48,22,48,D3,FC,00,00,00,40
253 FU  DATA D1,FC,00,00,00,44,20,3C,00,00
254 AM  DATA 00,0F,22,D8,51,C8,FF,FC,42,AA
255 dD  DATA 00,7C,4E,75,20,79,00,00,3B,10
256 si  DATA 61,00,00,28,20,79,00,00,3B,14
257 Xi  DATA 61,00,00,1E,20,79,00,00,3B,18
258 ty  DATA 61,00,00,14,20,79,00,00,3B,1C
259 Tr  DATA 61,00,00,0A,61,00,1E,5A,60,00
260 8J  DATA FD,DA,24,48,22,48,D3,FC,00,00
261 Bm  DATA 00,3C,D1,FC,00,00,00,40,20,3C
262 zk  DATA 00,00,00,0F,21,21,51,C8,FF,FC
263 VJ  DATA 42,92,4E,75,20,79,00,00,3B,10
264 0q  DATA 61,00,00,28,20,79,00,00,3B,14
265 fq  DATA 61,00,00,1E,20,79,00,00,3B,18
266 16  DATA 61,00,00,14,20,79,00,00,3B,1C
267 Mf  DATA 61,00,00,0A,61,00,1E,0A,60,00
268 5U  DATA FD,8A,20,3C,00,00,00,1F,22,10
269 8J  DATA E3,49,20,C1,51,C8,FF,F8,4E,75
270 tG  DATA 20,79,00,00,3B,10,61,00,00,28
271 51  DATA 20,79,00,00,3B,14,61,00,00,1E
272 vL  DATA 20,79,00,00,3B,18,61,00,00,14
273 SF  DATA 20,79,00,00,3B,1C,61,00,00,0A
274 p1  DATA 61,00,1D,C8,60,00,FD,48,20,3C
275 09  DATA 00,00,00,1F,22,10,48,41,E2,49
276 uL  DATA 48,41,20,C1,51,C8,FF,F4,4E,75
277 xr  DATA 2C,79,00,00,3A,46,20,79,00,00
278 Me  DATA 3A,66,43,F9,00,00,34,36,20,3C
279 UP  DATA 00,00,01,09,22,3C,00,00,00,50
280 4w  DATA 4E,AE,FF,8E,33,FC,00,00,00,00
281 RF  DATA 3A,7C,60,00,FD,04,41,F9,00,00
282 Vi  DATA 2D,96,2C,79,00,00,3A,46,4E,AE
283 Qt  DATA FF,34,23,C0,00,00,3A,56,67,00
284 kJ  DATA FC,EA,20,40,23,E8,00,32,00,00
285 bU  DATA 3A,6A,22,68,00,32,30,3C,00,01
286 W9  DATA 2C,79,00,00,3A,3E,4E,AE,FE,AA
287 VT  DATA 22,79,00,00,3A,6A,30,3C,00,01
288 wx  DATA 4E,AE,FE,A4,22,79,00,00,3A,6A
289 M9  DATA 30,3C,00,00,32,3C,00,00,34,3C
290 po  DATA 00,DF,36,3C,00,DF,4E,AE,FE,CE
291 3N  DATA 23,F9,00,00,3A,6A,00,00,3A,6E
292 EZ  DATA 4B,F9,00,00,3B,77,3A,3C,00,05
293 BE  DATA 3C,3C,00,5C,3E,3C,00,5B,4E,B9
294 4p  DATA 00,00,04,18,4B,F9,00,00,3B,7C
295 u2  DATA 3A,3C,00,09,3C,3C,00,4C,3E,3C
296 qi  DATA 00,6D,4E,B9,00,00,04,18,4B,F9
297 1s  DATA 00,00,3C,13,3A,3C,00,07,3C,3C
298 6I  DATA 00,54,3E,3C,00,7F,4E,B9,00,00
299 9D  DATA 04,18,30,3C,00,02,38,3C,00,46
300 I0  DATA 3A,3C,00,50,3C,3C,00,9A,3E,3C
301 E8  DATA 00,60,4E,B9,00,00,04,88,30,3C
302 rs  DATA 00,02,38,3C,00,46,3A,3C,00,62
303 jG  DATA 3C,3C,00,9A,3E,3C,00,72,4E,B9
304 oT  DATA 00,00,04,88,30,3C,00,02,38,3C
305 nw  DATA 00,46,3A,3C,00,74,3C,3C,00,9A
306 IT  DATA 3E,3C,00,84,4E,B9,00,00,04,88
307 1t  DATA 08,39,00,06,00,BF,E0,01,67,F6
308 kr  DATA 08,39,00,06,00,BF,E0,01,66,F6
309 rg  DATA 22,79,00,00,3A,56,3A,29,00,0E
310 fs  DATA 3C,29,00,0C,BA,7C,00,46,65,00
311 CX  DATA 00,1E,BC,7C,00,50,65,00,00,16
312 Vw  DATA BA,7C,00,9A,62,00,00,0E,BC,7C
313 D0  DATA 00,60,62,00,00,06,60,00,00,52
314 zd  DATA BA,7C,00,46,65,00,00,1E,BC,7C
315 Jr  DATA 00,62,65,00,00,16,BA,7C,00,9A
316 1Y  DATA 62,00,00,0E,BC,7C,00,72,62,00
317 cq  DATA 00,06,60,00,04,F4,BA,7C,00,46
318 gJ  DATA 65,9A,BC,7C,00,74,65,94,BA,7C
319 g3  DATA 00,9A,62,8E,BC,7C,00,84,62,88
320 eY  DATA 2C,79,00,00,3A,46,20,79,00,00
321 1t  DATA 3A,56,4E,AE,FF,B8,60,00,FB,70
322 ga  DATA 2C,79,00,00,3A,46,20,79,00,00
323 EZ  DATA 3A,56,4E,AE,FF,B8,20,79,00,00
324 04  DATA 3A,62,30,3C,00,1F,32,3C,00,0D
325 Fm  DATA 34,01,36,01,2C,79,00,00,3A,3E
326 Rr  DATA 4E,AE,FE,E0,20,79,00,00,3A,62
327 I4  DATA 30,3C,00,1E,32,3C,00,03,34,01
328 MR  DATA 36,01,4E,AE,FE,E0,2C,79,00,00
329 i8  DATA 3A,46,23,FC,00,00,2D,F6,00,00
330 xo  DATA 2D,A8,41,F9,00,00,2D,96,4E,AE
331 qW  DATA FF,34,23,C0,00,00,3A,56,20,40
332 xf  DATA 23,E8,00,32,00,00,3A,6A,23,E8
333 6F  DATA 00,32,00,00,3A,6E,22,68,00,32
334 er  DATA 30,3C,00,01,2C,79,00,00,3A,3E
335 7L  DATA 4E,AE,FE,AA,22,79,00,00,3A,6A
336 T8  DATA 30,3C,00,01,4E,AE,FE,A4,22,79
337 6A  DATA 00,00,3A,6A,30,3C,00,00,32,3C
338 Jd  DATA 00,00,34,3C,00,DF,36,3C,00,54
339 Vf  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
340 kh  DATA 30,3C,00,00,32,3C,00,55,34,3C
341 m5  DATA 00,5D,36,3C,00,61,4E,AE,FE,CE
342 u0  DATA 22,79,00,00,3A,6A,30,3C,00,81
343 au  DATA 32,3C,00,55,34,3C,00,DF,36,3C
344 1S  DATA 00,61,4E,AE,FE,CE,22,79,00,00
345 w4  DATA 3A,6A,30,3C,00,00,32,3C,00,61
346 cD  DATA 34,3C,00,DF,36,3C,00,72,4E,AE
347 NW  DATA FE,CE,22,79,00,00,3A,6A,30,3C
348 cT  DATA 00,00,32,3C,00,73,34,3C,00,24
349 WT  DATA 36,3C,00,7F,4E,AE,FE,CE,22,79
350 hB  DATA 00,00,3A,6A,30,3C,00,B8,32,3C
351 xW  DATA 00,73,34,3C,00,DF,36,3C,00,7F
352 is  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
353 9P  DATA 30,3C,00,00,32,3C,00,7F,34,3C
354 rq  DATA 00,DF,36,3C,00,DF,4E,AE,FE,CE
355 vC  DATA 4B,F9,00,00,3B,85,3A,3C,00,11
356 jp  DATA 3C,3C,00,27,3E,3C,00,4F,4E,B9
357 Ak  DATA 00,00,04,18,4B,F9,00,00,3B,96
358 3x  DATA 3A,3C,00,09,3C,3C,00,27,3E,3C
359 ff  DATA 00,6B,4E,B9,00,00,04,18,4B,F9
360 Iu  DATA 00,00,3B,9F,3A,3C,00,06,3C,3C
361 YB  DATA 00,58,3E,3C,00,91,4E,B9,00,00
362 47  DATA 04,18,30,3C,00,02,38,3C,00,54
363 uY  DATA 3A,3C,00,87,3C,3C,00,8C,3E,3C
364 Gf  DATA 00,96,4E,B9,00,00,04,88,30,39
365 39  DATA 00,00,2E,82,08,00,00,07,66,00
366 Wv  DATA 00,36,08,39,00,06,00,BF,E0,01
367 Hh  DATA 66,E8,22,79,00,00,3A,56,3A,29
368 30  DATA 00,0E,3C,29,00,0C,BA,7C,00,54
369 t2  DATA 65,D4,BC,7C,00,87,65,CE,BA,7C
370 f6  DATA 00,8C,62,C8,BC,7C,00,96,62,C2
371 8R  DATA 60,00,00,0E,30,39,00,00,2E,82
372 g0  DATA 08,00,00,07,66,F4,0C,B9,00,00
373 JV  DATA 00,00,00,00,2E,62,66,00,00,0C
```

Listing 1. Der Quellcode des Object-Editors als DATA-Lader. Bitte mit dem Checksummer auf Seite 159 eingeben.

```
374 K9  DATA 23,FC,00,00,00,01,00,00,2E,62
375 XR  DATA 2C,79,00,00,3A,46,20,79,00,00
376 kI  DATA 3A,56,4E,AE,FF,B8,23,FC,00,00
377 gl  DATA 00,00,00,00,2D,A8,2C,79,00,00
378 If  DATA 3A,3E,20,79,00,00,3A,62,43,F9
379 oE  DATA 00,00,32,6E,30,3C,00,20,4E,AE
380 mB  DATA FF,40,2C,79,00,00,3A,42,24,3C
381 vR  DATA 00,00,03,ED,22,3C,00,00,2E,EA
382 7m  DATA 4E,AE,FF,E2,23,C0,00,00,3A,86
383 N1  DATA 67,00,01,06,22,00,26,3C,00,00
384 hb  DATA 00,04,24,3C,00,00,00,00,4E,AE
385 CJ  DATA FF,D6,67,00,00,CC,0C,B9,41,4F
386 jK  DATA 45,20,00,00,00,00,66,00,00,F6
387 k5  DATA 22,39,00,00,3A,86,26,3C,00,00
388 pv  DATA 00,02,24,3C,00,00,00,04,4E,AE
389 cI  DATA FF,D6,67,00,00,A4,20,39,00,00
390 sp  DATA 2E,62,D0,79,00,00,00,04,B0,B9
391 xu  DATA 00,00,3A,78,62,00,00,8E,22,39
392 Iy  DATA 00,00,3A,86,26,3C,00,00,00,20
393 Sr  DATA 24,3C,00,00,32,8E,4E,AE,FF,D6
394 k3  DATA 67,00,00,74,42,82,24,39,00,00
395 Ad  DATA 2E,62,94,BC,00,00,00,01,C4,FC
396 mo  DATA 02,80,D4,B9,00,00,3A,4A,42,83
397 51  DATA 36,39,00,00,00,04,C6,FC,02,80
398 xy  DATA 22,39,00,00,3A,86,4E,AE,FF,D6
399 iK  DATA 67,00,00,42,4E,B9,00,00,2A,F2
400 dl  DATA 2C,79,00,00,3A,3E,20,79,00,00
401 ys  DATA 3A,62,43,F9,00,00,32,6E,30,3C
402 eV  DATA 00,20,4E,AE,FF,40,4E,B9,00,00
403 x5  DATA 27,A6,4E,B9,00,00,05,C2,2C,79
404 sn  DATA 00,00,3A,42,22,39,00,00,3A,86
405 cI  DATA 4E,AE,FF,DC,60,00,F8,2A,20,79
406 i4  DATA 00,00,2D,B4,2C,79,00,00,3A,46
407 pn  DATA 4E,AE,FF,A0,2C,79,00,00,3A,42
408 AO  DATA 22,39,00,00,3A,86,4E,AE,FF,DC
409 gS  DATA 60,00,F8,06,20,79,00,00,2D,B4
410 Ud  DATA 2C,79,00,00,3A,46,4E,AE,FF,A0
411 10  DATA 60,00,F7,F2,0C,B9,46,4F,52,4D
412 6F  DATA 00,00,00,00,66,BC,22,39,00,00
413 VQ  DATA 3A,86,26,3C,00,00,00,04,24,3C
414 gN  DATA 00,00,00,00,4E,AE,FF,D6,67,A4
415 Gb  DATA 0C,B9,00,00,04,00,00,00,00,00
416 Bk  DATA 64,98,22,39,00,00,3A,86,26,3C
417 SE  DATA 00,00,04,00,24,3C,00,00,00,00
418 04  DATA 4E,AE,FF,D6,41,F9,00,00,00,00
419 iY  DATA 20,18,B0,BC,49,4C,42,4D,66,00
420 eu  DATA FF,72,20,18,B0,BC,42,4D,48,44
421 Iw  DATA 67,00,00,3E,B1,FC,00,00,04,00
422 Sn  DATA 66,EC,60,00,FF,5A,20,18,B0,BC
423 1j  DATA 43,4D,41,50,67,00,00,50,B1,FC
424 Jb  DATA 00,00,04,00,66,EC,60,00,FF,42
425 xS  DATA 20,18,B0,BC,42,4F,44,59,67,00
426 jH  DATA 00,6A,B1,FC,00,00,04,00,66,EC
427 du  DATA 60,00,FF,2A,D1,FC,00,00,00,04
428 zz  DATA 20,18,B0,BC,00,20,00,20,66,00
429 OE  DATA FF,18,D1,FC,00,00,00,04,10,18
430 w1  DATA B0,3C,00,04,66,00,FF,08,D1,FC
431 Hy  DATA 00,00,00,03,60,A6,D1,FC,00,00
432 Gi  DATA 00,04,43,F9,00,00,32,8E,20,3C
433 Ow  DATA 00,00,00,0F,42,41,42,42,12,18
434 34  DATA E9,49,84,41,42,41,12,18,84,41
435 n4  DATA 42,41,12,18,E8,49,84,41,32,C2
436 lm  DATA 51,C8,FF,E4,60,8C,D1,FC,00,00
437 I6  DATA 00,04,20,39,00,00,2E,62,90,BC
438 Mk  DATA 00,00,00,01,C0,FC,02,80,D0,B9
439 4f  DATA 00,00,3A,4A,22,40,24,40,26,40
440 Pn  DATA 28,40,D5,FC,00,00,00,80,D7,FC
441 Nv  DATA 00,00,01,00,D9,FC,00,00,01,80
442 Th  DATA 20,3C,00,00,00,1F,22,D8,24,D8
443 21  DATA 26,D8,28,D8,51,C8,FF,F6,60,00
444 Xh  DATA FE,42,2C,79,00,00,3A,46,20,79
445 H8  DATA 00,00,3A,56,4E,AE,FF,B8,20,79
446 q2  DATA 00,00,3A,62,30,3C,00,1F,32,3C
447 EY  DATA 00,0D,34,01,36,01,2C,79,00,00
448 AP  DATA 3A,3E,4E,AE,FE,E0,20,79,00,00
449 zN  DATA 3A,62,30,3C,00,1E,32,3C,00,03
450 Xn  DATA 34,01,36,01,4E,AE,FE,E0,2C,79
451 tf  DATA 00,00,3A,46,23,FC,00,00,2F,3E
452 yd  DATA 00,00,2D,A8,41,F9,00,00,2D,96
453 OC  DATA 4E,AE,FF,34,23,C0,00,00,3A,56
454 5p  DATA 20,40,23,E8,00,32,00,00,3A,6A
455 Ci  DATA 23,E8,00,32,00,00,3A,6E,22,68
456 oE  DATA 00,32,30,3C,00,01,2C,79,00,00
457 CW  DATA 3A,3E,4E,AE,FE,AA,22,79,00,00
458 Id  DATA 3A,6A,30,3C,00,01,4E,AE,FE,A4
459 C9  DATA 22,79,00,00,3A,6A,30,3C,00,00
460 p0  DATA 32,3C,00,00,34,3C,00,DF,36,3C
461 7L  DATA 00,54,4E,AE,FE,CE,22,79,00,00
462 5G  DATA 3A,6A,30,3C,00,00,32,3C,00,55
463 Ca  DATA 34,3C,00,35,36,3C,00,61,4E,AE
464 GP  DATA FE,CE,22,79,00,00,3A,6A,30,3C
465 Hk  DATA 00,59,32,3C,00,55,34,3C,00,85
466 dJ  DATA 36,3C,00,61,4E,AE,FE,CE,22,79
467 b5  DATA 00,00,3A,6A,30,3C,00,A9,32,3C
468 7A  DATA 00,55,34,3C,00,DF,36,3C,00,61
469 bl  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
470 dX  DATA 30,3C,00,00,32,3C,00,61,34,3C
471 Mo  DATA 00,DF,36,3C,00,72,4E,AE,FE,CE
472 PM  DATA 22,79,00,00,3A,6A,30,3C,00,00
473 la  DATA 32,3C,00,73,34,3C,00,1D,36,3C
474 EK  DATA 00,7F,4E,AE,FE,CE,22,79,00,00
475 tN  DATA 3A,6A,30,3C,00,B1,32,3C,00,73
476 Ov  DATA 34,3C,00,DF,36,3C,00,7F,4E,AE
477 Tc  DATA FE,CE,22,79,00,00,3A,6A,30,3C
478 Fz  DATA 00,00,32,3C,00,7F,34,3C,00,DF
479 Y2  DATA 36,3C,00,DF,4E,AE,FE,CE,4B,F9
480 NB  DATA 00,00,3B,A5,3A,3C,00,14,3C,3C
481 NI  DATA 00,1F,3E,3C,00,4F,4E,B9,00,00
482 bG  DATA 04,18,4B,F9,00,00,3B,B9,3A,3C
483 Gc  DATA 00,04,3C,3C,00,5F,3E,3C,00,5D
484 ji  DATA 4E,B9,00,00,04,18,4B,F9,00,00
485 yS  DATA 3B,96,3A,3C,00,09,3C,3C,00,1F
486 0Q  DATA 3E,3C,00,6B,4E,B9,00,00,04,18
487 Fs  DATA 4B,F9,00,00,3B,9F,3A,3C,00,06
488 4j  DATA 3C,3C,00,58,3E,3C,00,91,4E,B9
489 7E  DATA 00,00,04,18,30,3C,00,02,38,3C
490 wJ  DATA 00,54,3A,3C,00,87,3C,3C,00,8C
491 Ra  DATA 3E,3C,00,96,4E,B9,00,00,04,88
492 mE  DATA 30,39,00,00,30,02,08,00,00,07
493 Qq  DATA 66,00,00,36,08,39,00,06,00,BF
494 JO  DATA E0,01,66,E8,22,79,00,00,3A,56
495 L5  DATA 3A,29,00,0E,3C,29,00,0C,BA,7C
496 oR  DATA 00,54,65,D4,BC,7C,00,87,65,CE
497 gT  DATA BA,7C,00,8C,62,C8,BC,7C,00,96
498 KZ  DATA 62,C2,60,00,00,0E,30,39,00,00
499 5N  DATA 30,02,08,00,00,07,66,F4,0C,B9
500 ui  DATA 00,00,00,00,00,00,2F,86,66,00
501 12  DATA 00,0C,23,FC,00,00,00,01,00,00
502 aF  DATA 2F,86,2C,79,00,00,3A,46,20,79
503 50  DATA 00,00,3A,56,4E,AE,FF,B8,23,FC
504 YO  DATA 00,00,00,00,00,00,2D,A8,2C,79
505 X1  DATA 00,00,3A,3E,20,79,00,00,3A,62
506 OU  DATA 43,F9,00,00,32,6E,30,3C,00,20
507 9v  DATA 4E,AE,FF,40,0C,B9,00,00,00,00
508 1K  DATA 00,00,2F,86,66,00,00,0C,23,FC
509 of  DATA 00,00,00,01,00,00,2F,86,0C,B9
510 RO  DATA 00,00,00,00,00,00,2F,E2,66,00
511 BC  DATA 00,0C,23,FC,00,00,00,01,00,00
512 Cy  DATA 2F,E2,20,39,00,00,2F,86,22,39
513 UT  DATA 00,00,2F,E2,B2,80,6C,00,00,0E
514 xI  DATA 23,C1,00,00,2F,86,23,C0,00,00
515 Ng  DATA 2F,E2,20,39,00,00,3A,78,B0,B9
516 40  DATA 00,00,2F,E2,65,00,01,04,2C,79
517 eB  DATA 00,00,3A,42,24,3C,00,00,03,EE
518 oZ  DATA 22,3C,00,00,2E,EA,4E,AE,FF,E2
519 4J  DATA 23,C0,00,00,3A,86,67,00,00,E4
520 I7  DATA 23,FC,41,4F,45,20,00,00,00,00
521 wn  DATA 22,00,24,3C,00,00,00,00,26,3C
522 k4  DATA 00,00,00,04,4E,AE,FF,D0,67,00
523 g2  DATA 00,A0,20,39,00,00,2F,E2,90,B9
524 3C  DATA 00,00,2F,86,D0,7C,00,01,33,C0
525 8x  DATA 00,00,00,00,22,39,00,00,3A,86
526 w7  DATA 24,3C,00,00,00,00,26,3C,00,00
527 1a  DATA 00,02,4E,AE,FF,D0,67,00,00,70
528 pI  DATA 22,39,00,00,3A,86,24,3C,00,00
529 kp  DATA 32,8E,26,3C,00,00,00,20,4E,AE
530 wl  DATA FF,D0,67,00,00,56,4E,B9,00,00
531 Mw  DATA 2B,20,24,39,00,00,2F,86,94,BC
532 Kq  DATA 00,00,00,01,C4,FC,02,80,D4,B9
533 Lr  DATA 00,00,3A,4A,26,39,00,00,2F,E2
534 Gc  DATA 96,B9,00,00,2F,86,D6,BC,00,00
535 H1  DATA 00,01,C6,FC,02,80,22,39,00,00
536 Dy  DATA 3A,86,4E,AE,FF,D0,67,00,00,16
537 Wg  DATA 4E,B9,00,00,2A,F2,22,39,00,00
538 2V  DATA 3A,86,4E,AE,FF,DC,60,00,F2,F6
539 dE  DATA 20,79,00,00,2D,B4,2C,79,00,00
540 41  DATA 3A,46,4E,AE,FF,A0,2C,79,00,00
541 Ut  DATA 3A,42,22,39,00,00,3A,86,4E,AE
542 fr  DATA FF,DC,60,00,F2,D2,20,79,00,00
543 Fj  DATA 2D,B4,2C,79,00,00,3A,46,4E,AE
544 Jj  DATA FF,A0,60,00,F2,BE,2C,79,00,00
545 Y7  DATA 3A,3E,43,F9,00,00,3A,98,20,3C
546 mp  DATA 00,00,00,10,4E,AE,FE,AA,43,F9
547 ze  DATA 00,00,3A,98,42,80,42,81,34,3C
548 qD  DATA 00,1F,36,3C,00,1F,4E,AE,FE,CE
549 hY  DATA 4E,B9,00,00,27,A6,60,00,F2,88
550 Pm  DATA 20,79,00,00,3B,10,61,00,00,28
551 bE  DATA 20,79,00,00,3B,14,61,00,00,1E
552 Rr  DATA 20,79,00,00,3B,18,61,00,00,14
553 yl  DATA 20,79,00,00,3B,1C,61,00,00,0A
554 ab  DATA 61,00,12,D8,60,00,F2,58,20,3C
555 k8  DATA 00,00,00,1F,22,10,24,10,48,42
556 xg  DATA C2,BC,00,00,FF,FF,C4,BC,00,00
557 I9  DATA FF,FF,42,83,42,84,2E,3C,00,00
558 zR  DATA 00,0F,0F,01,67,00,00,10,46,47
559 La  DATA CE,7C,00,0F,0F,C3,46,47,CE,7C
560 Hh  DATA 00,0F,51,CF,FF,EA,2E,3C,00,00
561 4X  DATA 00,0F,0F,02,67,00,00,10,46,47
562 Rh  DATA CE,7C,00,0F,0F,C4,46,47,CE,7C
563 3s  DATA 00,0F,51,CF,FF,EA,48,43,86,84
564 oK  DATA 20,C3,51,C8,FF,A6,4E,75,20,79
565 yu  DATA 00,00,3B,10,61,00,00,28,20,79
566 u6  DATA 00,00,3B,14,61,00,00,1E,20,79
567 xw  DATA 00,00,3B,18,61,00,00,14,20,79
568 MD  DATA 00,00,3B,1C,61,00,00,0A,61,00
569 f8  DATA 12,44,60,00,F1,C4,22,48,D3,FC
570 tN  DATA 00,00,00,80,20,3C,00,00,00,0F
571 EG  DATA 22,10,24,21,22,81,20,C2,51,C8
572 Tq  DATA FF,F6,4E,75,20,79,00,00,3B,10
573 zp  DATA 61,00,00,28,20,79,00,00,3B,14
574 ep  DATA 61,00,00,1E,20,79,00,00,3B,18
575 05  DATA 61,00,00,14,20,79,00,00,3B,1C
576 th  DATA 61,00,00,0A,61,00,11,F8,60,00
577 Rd  DATA F1,78,43,F9,00,00,00,00,24,48
578 3s  DATA 2C,3C,00,00,00,1F,2E,3C,00,00
579 CY  DATA 00,1F,42,83,20,4A,20,18,0D,00
580 Ae  DATA 67,00,00,10,46,47,CE,7C,00,1F
581 V2  DATA 0F,C3,46,47,CE,7C,00,1F,51,CF
582 qw  DATA FF,E8,22,C3,51,CE,FF,D8,43,F9
583 re  DATA 00,00,00,00,20,4A,2C,3C,00,00
584 6S  DATA 00,1F,20,D9,51,CE,FF,FC,4E,75
585 FI  DATA 20,79,00,00,3A,62,30,3C,00,1F
586 oX  DATA 32,3C,00,0D,34,01,36,01,2C,79
587 Ob  DATA 00,00,3A,3E,4E,AE,FE,E0,20,79
588 4F  DATA 00,00,3A,62,30,3C,00,1E,32,3C
589 ea  DATA 00,03,34,01,36,01,4E,AE,FE,E0
590 fm  DATA 2C,79,00,00,3A,46,23,FC,00,00
591 T2  DATA 31,5A,00,00,2D,A8,41,F9,00,00
592 cb  DATA 2D,96,4E,AE,FF,34,23,C0,00,00
593 fL  DATA 3A,56,20,40,23,E8,00,32,00,00
594 KM  DATA 3A,6A,23,E8,00,32,00,00,3A,6E
595 dE  DATA 22,68,00,32,30,3C,00,01,2C,79
596 Vx  DATA 00,00,3A,3E,4E,AE,FE,AA,22,79
597 81  DATA 00,00,3A,6A,30,3C,00,01,4E,AE
598 cp  DATA FE,A4,22,79,00,00,3A,6A,30,3C
599 uB  DATA 00,00,32,3C,00,00,34,3C,00,DF
600 na  DATA 36,3C,00,54,4E,AE,FE,CE,22,79
601 MQ  DATA 00,00,3A,6A,30,3C,00,00,32,3C
602 uP  DATA 00,55,34,3C,00,35,36,3C,00,61
603 lv  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
604 mY  DATA 30,3C,00,59,32,3C,00,55,34,3C
605 p9  DATA 00,85,36,3C,00,61,4E,AE,FE,CE
606 O1  DATA 22,79,00,00,3A,6A,30,3C,00,A9
607 qA  DATA 32,3C,00,55,34,3C,00,DF,36,3C
608 Hi  DATA 00,61,4E,AE,FE,CE,22,79,00,00
609 CK  DATA 3A,6A,30,3C,00,00,32,3C,00,61
610 Dt  DATA 34,3C,00,DF,36,3C,00,68,4E,AE
611 dm  DATA FE,CE,22,79,00,00,3A,6A,30,3C
612 rx  DATA 00,00,32,3C,00,69,34,3C,00,79
613 Eu  DATA 36,3C,00,75,4E,AE,FE,CE,22,79
614 Ip  DATA 00,00,3A,6A,30,3C,00,9D,32,3C
615 LE  DATA 00,69,34,3C,00,DF,36,3C,00,75
616 y8  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
617 JI  DATA 30,3C,00,00,32,3C,00,75,34,3C
618 76  DATA 00,DF,36,3C,00,DF,4E,AE,FE,CE
619 7e  DATA 4B,F9,00,00,3C,4F,3A,3C,00,13
620 rt  DATA 3C,3C,00,23,3E,3C,00,4F,4E,B9
621 F1  DATA 00,00,04,18,4B,F9,00,00,3B,B9
622 GN  DATA 3A,3C,00,04,3C,3C,00,5F,3E,3C
623 2t  DATA 00,5D,4E,B9,00,00,04,18,4B,F9
624 gB  DATA 00,00,3C,62,3A,3C,00,06,3C,3C
625 wb  DATA 00,3F,3E,3C,00,71,4E,B9,00,00
```

```
626 K3  DATA 04,18,4B,F9,00,00,3B,9F,3A,3C
627 pa  DATA 00,06,3C,3C,00,58,3E,3C,00,91
628 78  DATA 4E,B9,00,00,04,18,30,3C,00,02
629 N1  DATA 38,3C,00,54,3A,3C,00,87,3C,3C
630 RU  DATA 00,8C,3E,3C,00,96,4E,B9,00,00
631 cx  DATA 04,88,30,39,00,00,32,1E,08,00
632 vq  DATA 00,07,66,00,00,36,08,39,00,06
633 3M  DATA 00,BF,E0,01,66,E8,22,79,00,00
634 AQ  DATA 3A,56,3A,29,00,0E,3C,29,00,0C
635 Ey  DATA BA,7C,00,54,65,D4,BC,7C,00,87
636 9M  DATA 65,CE,BA,7C,00,8C,62,C8,BC,7C
637 rF  DATA 00,96,62,C2,60,00,00,0E,30,39
638 xw  DATA 00,00,32,1E,08,00,00,07,66,F4
639 nh  DATA 2C,79,00,00,3A,46,20,79,00,00
640 OY  DATA 3A,56,4E,AE,FF,B8,23,FC,00,00
641 w1  DATA 00,00,00,00,2D,A8,2C,79,00,00
642 Yv  DATA 3A,3E,20,79,00,00,3A,62,43,F9
643 4U  DATA 00,00,32,6E,30,3C,00,20,4E,AE
644 k2  DATA FF,40,4A,B9,00,00,31,A2,62,00
645 LM  DATA 00,0C,23,FC,00,00,00,01,00,00
646 ry  DATA 31,A2,4A,B9,00,00,31,FE,62,00
647 NO  DATA 00,0C,23,FC,00,00,00,01,00,00
648 B1  DATA 31,FE,4A,B9,00,00,32,5A,63,00
649 JO  DATA 01,00,20,39,00,00,31,A2,D0,B9
650 x0  DATA 00,00,32,5A,90,BC,00,00,00,01
651 Jm  DATA B0,B9,00,00,3A,78,62,00,00,EE
652 3E  DATA 22,39,00,00,31,FE,D2,B9,00,00
653 SQ  DATA 32,5A,92,BC,00,00,00,01,B2,B9
654 hR  DATA 00,00,3A,78,62,00,00,D2,B2,80
655 c8  DATA 67,00,00,C2,B0,B9,00,00,31,FE
656 aR  DATA 62,00,00,4C,20,39,00,00,32,5A
657 dz  DATA C0,FC,00,A0,90,BC,00,00,00,01
658 pb  DATA 22,39,00,00,31,A2,92,BC,00,00
659 CP  DATA 00,01,C2,FC,02,80,D2,B9,00,00
660 JQ  DATA 3A,4A,20,41,22,39,00,00,31,FE
661 zV  DATA 92,BC,00,00,00,01,C2,FC,02,80
662 40  DATA D2,B9,00,00,3A,4A,22,41,22,D8
663 av  DATA 51,C8,FF,FC,60,00,00,6E,20,39
664 Fk  DATA 00,00,31,A2,B0,B9,00,00,31,FE
665 PF  DATA 62,A8,20,39,00,00,32,5A,C0,FC
666 95  DATA 00,A0,90,BC,00,00,00,01,22,39
667 CR  DATA 00,00,31,A2,D2,B9,00,00,32,5A
668 6c  DATA 92,BC,00,00,00,01,C2,FC,02,80
669 yA  DATA D2,B9,00,00,3A,4A,20,41,22,39
670 LZ  DATA 00,00,31,FE,D2,B9,00,00,32,5A
671 9f  DATA 92,BC,00,00,00,01,C2,FC,02,80
672 xJ  DATA D2,B9,00,00,3A,4A,22,41,D1,FC
673 Tt  DATA 00,00,00,04,D3,FC,00,00,00,04
674 HA  DATA 23,20,51,C8,FF,FC,4E,B9,00,00
675 yM  DATA 27,A6,60,00,ED,A0,2C,79,00,00
676 OO  DATA 3A,46,20,79,00,00,2D,84,4E,AE
677 ZV  DATA FF,A0,60,00,ED,8C,41,F9,00,00
678 t6  DATA 2D,96,2C,79,00,00,3A,46,4E,AE
679 FW  DATA FF,34,4A,80,67,00,ED,76,23,C0
680 UK  DATA 00,00,3A,56,20,40,23,E8,00,32
681 WW  DATA 00,00,3A,6A,22,68,00,32,30,3C
682 dt  DATA 00,01,2C,79,00,00,3A,3E,4E,AE
683 Fc  DATA FE,AA,22,79,00,00,3A,6A,30,3C
684 8A  DATA 00,01,4E,AE,FE,A4,22,79,00,00
685 Tt  DATA 3A,6A,42,40,42,41,34,3C,00,DF
686 Da  DATA 36,3C,00,DF,4E,AE,FE,CE,23,F9
687 6U  DATA 00,00,3A,6A,00,00,3A,6E,4B,F9
688 ER  DATA 00,00,3B,BD,3A,3C,00,0C,3C,3C
689 or  DATA 00,40,3E,3C,00,50,4E,B9,00,00
690 0g  DATA 04,18,4B,F9,00,00,3B,C9,3A,3C
691 pv  DATA 00,0B,3C,3C,00,44,3E,3C,00,5A
692 54  DATA 4E,B9,00,00,04,18,4B,F9,00,00
693 K4  DATA 3B,D4,3A,3C,00,0F,3C,3C,00,34
694 4L  DATA 3E,3C,00,69,4E,B9,00,00,04,18
695 M8  DATA 4B,F9,00,00,3B,E3,3A,3C,00,0F
696 8r  DATA 3C,3C,00,34,3E,3C,00,73,4E,B9
697 At  DATA 00,00,04,18,4B,F9,00,00,3B,F2
698 93  DATA 3A,3C,00,11,3C,3C,00,2C,3E,3C
699 1p  DATA 00,82,4E,B9,00,00,04,18,4B,F9
700 m0  DATA 00,00,3B,9F,3A,3C,00,06,3C,3C
701 Hz  DATA 00,58,3E,3C,00,96,4E,B9,00,00
702 Yb  DATA 04,18,30,3C,00,02,38,3C,00,54
703 mc  DATA 3A,3C,00,8C,3C,3C,00,8C,3E,3C
704 sp  DATA 00,9B,4E,B9,00,00,04,88,08,39
705 1V  DATA 00,06,00,BF,E0,01,66,F6,22,79
706 Uy  DATA 00,00,3A,56,3A,29,00,0E,3C,29
707 Jc  DATA 00,0C,BA,7C,00,54,65,E2,BC,7C
708 mc  DATA 00,8C,65,DC,BA,7C,00,8C,62,D6
709 us  DATA BC,7C,00,9B,62,D0,2C,79,00,00
710 iW  DATA 3A,46,20,79,00,00,3A,56,4E,AE
711 CM  DATA FF,B8,60,00,EC,38,41,F9,00,00
712 3k  DATA 2D,96,23,FC,00,00,30,A2,00,00
713 IV  DATA 2D,A8,2C,79,00,00,3A,46,4E,AE
714 1h  DATA FF,34,23,C0,00,00,3A,56,20,40
715 8q  DATA 23,E8,00,32,00,00,3A,6A,23,E8
716 HQ  DATA 00,32,00,00,3A,6E,22,68,00,32
717 p2  DATA 30,3C,00,01,2C,79,00,00,3A,3E
718 IW  DATA 4E,AE,FE,AA,22,79,00,00,3A,6A
719 aD  DATA 30,3C,00,01,4E,AE,FE,A4,20,79
720 GS  DATA 00,00,3A,62,30,3C,00,1F,32,3C
721 ey  DATA 00,0D,34,01,36,01,2C,79,00,00
722 ap  DATA 3A,3E,4E,AE,FE,E0,20,79,00,00
723 Pn  DATA 3A,62,30,3C,00,1E,32,3C,00,03
724 PO  DATA 34,01,36,01,4E,AE,FE,E0,22,79
725 MQ  DATA 00,00,3A,6A,30,3C,00,00,32,3C
726 wX  DATA 00,00,34,3C,00,DF,36,3C,00,5E
727 1v  DATA 4E,AE,FE,CE,22,79,00,00,3A,6A
728 6K  DATA 30,3C,00,00,32,3C,00,5F,34,3C
729 F1  DATA 00,35,36,3C,00,6B,4E,AE,FE,CE
730 cn  DATA 22,79,00,00,3A,6A,30,3C,00,59
731 iq  DATA 32,3C,00,5F,34,3C,00,85,36,3C
732 vG  DATA 00,6B,4E,AE,FE,CE,22,79,00,00
733 og  DATA 3A,6A,30,3C,00,A9,32,3C,00,5F
734 rh  DATA 34,3C,00,DF,36,3C,00,6B,4E,AE
735 dm  DATA FE,CE,22,79,00,00,3A,6A,30,3C
736 Bq  DATA 00,00,32,3C,00,6B,34,3C,00,DF
737 iC  DATA 36,3C,00,DF,4E,AE,FE,CE,4B,F9
738 RU  DATA 00,00,3C,1A,3A,3C,00,14,3C,3C
739 wf  DATA 00,1F,3E,3C,00,59,4E,B9,00,00
740 1Q  DATA 04,18,4B,F9,00,00,3B,B9,3A,3C
741 MV  DATA 00,04,3C,3C,00,5F,3E,3C,00,66
742 ts  DATA 4E,B9,00,00,04,18,4B,F9,00,00
743 fu  DATA 3B,9F,3A,3C,00,06,3C,3C,00,58
744 K1  DATA 3E,3C,00,7D,4E,B9,00,00,04,18
745 BS  DATA 30,3C,00,02,38,3C,00,54,3A,3C
746 fx  DATA 00,73,3C,3C,00,8C,3E,3C,00,82
747 Tw  DATA 4E,B9,00,00,04,88,08,39,00,06
748 q8  DATA 00,BF,E0,01,66,F6,22,79,00,00
749 1H  DATA 3A,56,3A,29,00,0E,3C,29,00,0C
750 dH  DATA BA,7C,00,54,65,E2,BC,7C,00,73
751 o9  DATA 65,DC,BA,7C,00,8C,62,D6,BC,7C
752 EA  DATA 00,82,62,D0,2C,79,00,00,3A,46
753 qq  DATA 20,79,00,00,3A,56,4E,AE,FF,B8
754 Q7  DATA 0C,B9,00,00,00,00,00,00,30,EA
755 Ms  DATA 66,00,00,0C,23,FC,00,00,00,01
756 WI  DATA 00,00,30,EA,20,39,00,00,3A,78
757 1D  DATA B0,B9,00,00,30,EA,62,00,00,0C
758 HN  DATA 23,F9,00,00,3A,78,00,00,30,EA
759 Yo  DATA 0C,B9,00,00,00,00,00,00,31,46
760 Rx  DATA 66,00,00,0C,23,FC,00,00,00,01
761 fN  DATA 00,00,31,46,20,39,00,00,3A,78
762 6r  DATA B0,B9,00,00,31,46,62,00,00,0C
763 P4  DATA 23,F9,00,00,3A,78,00,00,31,46
764 Hn  DATA 20,39,00,00,30,EA,22,39,00,00
765 pN  DATA 31,46,B2,80,64,00,00,0E,23,C1
766 8j  DATA 00,00,30,EA,23,C0,00,00,31,46
767 yI  DATA 23,F9,00,00,30,EA,00,00,3A,8E
768 HU  DATA 41,F9,00,00,2D,96,23,FC,00,00
769 05  DATA 00,00,00,00,2D,A8,2C,79,00,00
770 yq  DATA 3A,46,4E,AE,FF,34,23,C0,00,00
771 XD  DATA 3A,56,20,40,23,E8,00,32,00,00
772 CE  DATA 3A,6A,23,E8,00,32,00,00,3A,6E
773 nJ  DATA 22,68,00,32,22,79,00,00,3A,6A
774 kx  DATA 30,3C,00,01,2C,79,00,00,3A,3E
775 qn  DATA 4E,AE,FE,AA,20,79,00,00,3A,62
776 kq  DATA 43,F9,00,00,32,6E,30,3C,00,20
777 VJ  DATA 4E,AE,FF,40,22,79,00,00,3A,6A
778 bG  DATA 30,3C,00,01,4E,AE,FE,A4,22,79
779 EI  DATA 00,00,3A,6A,30,3C,00,00,32,3C
780 r1  DATA 00,00,34,3C,00,DF,36,3C,00,DF
781 zD  DATA 4E,AE,FE,CE,2C,79,00,00,3A,46
782 xq  DATA 20,79,00,00,3A,6A,43,F9,00,00
783 12  DATA 35,9E,30,3C,00,58,32,3C,00,57
784 BM  DATA 4E,AE,FF,8E,2C,79,00,00,3A,3E
785 9v  DATA 4B,F9,00,00,3C,2E,3A,3C,00,0F
786 fe  DATA 3C,3C,00,34,3E,3C,00,4F,4E,B9
787 OE  DATA 00,00,04,18,4B,F9,00,00,3B,9F
788 61  DATA 3A,3C,00,06,3C,3C,00,58,3E,3C
789 s1  DATA 00,7D,4E,B9,00,00,04,18,4B,F9
790 SO  DATA 00,00,3C,3D,3A,3C,00,03,3C,3C
791 s5  DATA 00,63,3E,3C,00,5D,4E,B9,00,00
792 03  DATA 04,18,30,3C,00,02,38,3C,00,54
793 cG  DATA 3A,3C,00,73,3C,3C,00,8C,3E,3C
794 XL  DATA 00,82,4E,B9,00,00,04,88,30,3C
795 EH  DATA 00,10,22,79,00,00,3A,6A,4E,AE
796 4R  DATA FE,AA,22,79,00,00,3A,6A,30,3C
797 wj  DATA 00,5F,32,3C,00,14,34,3C,00,7E
798 LF  DATA 36,3C,00,33,4E,AE,FE,CE,30,3C
799 Ob  DATA 00,02,38,3C,00,5E,3A,3C,00,13
800 eM  DATA 3C,3C,00,7F,3E,3C,00,34,4E,B9
801 tZ  DATA 00,00,04,88,30,3C,00,03,38,3C
802 WM  DATA 00,5D,3A,3C,00,12,3C,3C,00,80
803 rN  DATA 3E,3C,00,35,4E,B9,00,00,04,88
804 Ai  DATA 30,3C,00,04,38,3C,00,5C,3A,3C
805 45  DATA 00,11,3C,3C,00,81,3E,3C,00,36
806 XZ  DATA 4E,B9,00,00,04,88,23,F9,00,00
807 A7  DATA 3B,10,00,00,00,00,23,F9,00,00
808 1W  DATA 3B,14,00,00,00,04,23,F9,00,00
809 Mv  DATA 3B,18,00,00,00,08,23,F9,00,00
810 y0  DATA 3B,1C,00,00,00,0C,23,F9,00,00
811 LK  DATA 3B,20,00,00,00,10,23,F9,00,00
812 1N  DATA 3A,92,00,00,3A,8A,2A,7C,00,DF
813 45  DATA F0,00,23,F8,00,6C,00,00,27,68
814 FM  DATA 21,FC,00,00,26,AC,00,6C,0C,79
815 61  DATA 00,01,00,00,3A,96,66,00,00,54
816 JT  DATA 20,39,00,DF,F0,04,C0,BC,00,01
817 tf  DATA FF,00,B0,BC,00,01,00,00,66,EC
818 Mh  DATA 2C,79,00,00,3A,3E,41,F9,00,00
819 4G  DATA 3A,98,22,79,00,00,3A,6A,42,80
820 Ec  DATA 42,81,24,3C,00,00,00,5F,26,3C
821 OC  DATA 00,00,00,14,28,3C,00,00,00,20
822 fF  DATA 2A,3C,00,00,00,20,2C,3C,00,00
823 Bi  DATA 00,C0,4E,AE,FD,D8,42,79,00,00
824 J8  DATA 3A,96,08,39,00,06,00,BF,E0,01
825 hv  DATA 66,98,22,79,00,00,3A,56,3A,29
826 1A  DATA 00,0E,3C,29,00,0C,BA,7C,00,58
827 pR  DATA 65,00,00,1E,BC,7C,00,57,65,00
828 sv  DATA 00,16,BA,7C,00,5F,62,00,00,0E
829 xY  DATA BC,7C,00,5E,62,00,00,06,60,00
830 Z6  DATA 0A,06,BA,7C,00,80,65,00,00,22
831 E1  DATA BC,7C,00,57,65,00,00,1A,BA,7C
832 AB  DATA 00,87,62,00,00,12,BC,7C,00,5E
833 7S  DATA 62,00,00,0A,60,00,0A,22,60,00
834 oI  DATA FF,40,BA,7C,00,54,65,00,FF,38
835 GI  DATA BC,7C,00,73,65,00,FF,30,BA,7C
836 ZN  DATA 00,8C,62,00,FF,28,BC,7C,00,82
837 da  DATA 62,00,FF,20,2A,7C,00,DF,F0,00
838 Vm  DATA 21,F9,00,00,27,68,00,6C,23,F9
839 0a  DATA 00,00,00,00,00,00,3B,10,23,F9
840 P7  DATA 00,00,00,04,00,00,3B,14,23,F9
841 oe  DATA 00,00,00,08,00,00,3B,18,23,F9
842 t5  DATA 00,00,00,0C,00,00,3B,1C,23,F9
843 1J  DATA 00,00,00,10,00,00,3B,20,2C,79
844 sq  DATA 00,00,3A,46,20,79,00,00,3A,56
845 t0  DATA 4E,AE,FF,B8,23,F9,00,00,3A,66
846 9Q  DATA 00,00,3A,6E,23,FC,00,00,00,00
847 EM  DATA 00,00,2D,A8,60,00,E6,E6,30,39
848 wb  DATA 00,00,3A,76,B0,7C,00,01,66,00
849 53  DATA 00,06,60,00,E6,D4,04,B9,00,00
850 yv  DATA 02,80,00,00,3B,10,04,B9,00,00
851 BC  DATA 02,80,00,00,3B,14,04,B9,00,00
852 OT  DATA 02,80,00,00,3B,18,04,B9,00,00
853 wC  DATA 02,80,00,00,3B,1C,04,B9,00,00
854 cQ  DATA 02,80,00,00,3B,20,04,79,00,01
855 2L  DATA 00,00,3A,76,4E,B9,00,00,2A,42
856 PX  DATA 4E,B9,00,00,27,A6,60,00,E6,8A
857 13  DATA 30,39,00,00,3A,76,B0,B9,00,00
858 iE  DATA 3A,78,66,00,00,06,60,00,E6,76
859 pA  DATA 06,B9,00,00,02,80,00,00,3B,10
860 AZ  DATA 06,B9,00,00,02,80,00,00,3B,14
861 Vy  DATA 06,B9,00,00,02,80,00,00,3B,18
862 P3  DATA 06,B9,00,00,02,80,00,00,3B,1C
863 xJ  DATA 06,B9,00,00,02,80,00,00,3B,20
864 Gq  DATA 06,79,00,01,00,00,3A,76,4E,B9
865 Z6  DATA 00,00,2A,42,4E,B9,00,00,27,A6
866 1S  DATA 60,00,E6,2C,42,80,30,39,00,00
867 hy  DATA 3A,72,C0,FC,00,02,20,40,D1,FC
868 14  DATA 00,00,32,8E,30,10,C0,7C,0F,00
869 3U  DATA E0,58,D0,7C,00,01,C0,7C,00,0F
870 GW  DATA E1,58,02,50,00,FF,81,50,4E,B9
871 Jt  DATA 00,00,27,6C,4E,B9,00,00,05,C2
872 NC  DATA 60,00,E5,F0,42,80,30,39,00,00
873 n4  DATA 3A,72,C0,FC,00,02,20,40,D1,FC
874 rA  DATA 00,00,32,8E,30,10,C0,7C,0F,00
```

Listing 1. (Fortsetzung)

```
875 cs  DATA E0,58,90,7C,00,01,C0,7C,00,0F
876 Mc  DATA E1,58,02,50,00,FF,81,50,4E,B9
877 Pz  DATA 00,00,27,6C,4E,B9,00,00,05,C2
878 DM  DATA 60,00,E5,B4,42,80,30,39,00,00
879 tA  DATA 3A,72,C0,FC,00,02,20,40,D1,FC
880 fy  DATA 00,00,32,8E,30,10,C0,7C,00,F0
881 dC  DATA E8,58,D0,7C,00,01,C0,7C,00,0F
882 uI  DATA E9,58,02,50,0F,0F,81,50,4E,B9
883 V5  DATA 00,00,27,6C,4E,B9,00,00,05,C2
884 NI  DATA 60,00,E5,78,42,80,30,39,00,00
885 zG  DATA 3A,72,C0,FC,00,02,20,40,D1,FC
886 14  DATA 00,00,32,8E,30,10,C0,7C,00,F0
887 Ca  DATA E8,58,90,7C,00,01,C0,7C,00,0F
888 00  DATA E9,58,02,50,0F,0F,81,50,4E,B9
889 bB  DATA 00,00,27,6C,4E,B9,00,00,05,C2
890 Rn  DATA 60,00,E5,3C,42,80,30,39,00,00
891 5M  DATA 3A,72,C0,FC,00,02,20,40,D1,FC
892 DW  DATA 00,00,32,8E,30,10,C0,7C,00,0F
893 fL  DATA D0,7C,00,01,C0,7C,00,0F,02,50
894 VW  DATA 0F,F0,81,50,4E,B9,00,00,27,6C
895 1h  DATA 4E,B9,00,00,05,C2,60,00,E5,04
896 CD  DATA 42,80,30,39,00,00,3A,72,C0,FC
897 iv  DATA 00,02,20,40,D1,FC,00,00,32,8E
898 Jj  DATA 30,10,C0,7C,00,0F,90,7C,00,01
899 sJ  DATA C0,7C,00,0F,02,50,0F,F0,81,50
900 bf  DATA 4E,B9,00,00,27,6C,4E,B9,00,00
901 vc  DATA 05,C2,60,00,E4,CC,2C,79,00,00
902 JD  DATA 3A,46,41,F9,00,00,2D,C6,4E,AE
903 4k  DATA FF,34,23,C0,00,00,3A,56,20,40
904 Bt  DATA 23,E8,00,32,00,00,3A,6A,23,E8
905 KT  DATA 00,32,00,00,3A,6E,22,68,00,32
906 s5  DATA 30,3C,00,01,2C,79,00,00,3A,3E
907 yv  DATA 4E,AE,FE,AA,20,79,00,00,3A,62
908 nr  DATA 30,3C,00,1F,32,3C,00,0D,34,01
909 tC  DATA 36,01,4E,AE,FE,E0,20,79,00,00
910 Qo  DATA 3A,62,30,3C,00,1E,32,3C,00,03
911 QP  DATA 34,01,36,01,4E,AE,FE,E0,22,79
912 Dn  DATA 00,00,3A,6A,30,3C,00,01,4E,AE
913 OH  DATA FE,A4,22,79,00,00,3A,6A,42,40
914 k8  DATA 42,41,34,3C,00,8F,36,3C,00,0C
915 DM  DATA 4E,AE,FE,CE,42,40,32,3C,00,0D
916 KE  DATA 34,3C,00,36,36,3C,00,18,22,79
917 m4  DATA 00,00,3A,6A,4E,AE,FE,CE,30,3C
918 Ig  DATA 00,5A,32,3C,00,0D,34,3C,00,8F
919 CG  DATA 36,3C,00,18,22,79,00,00,3A,6A
920 TS  DATA 4E,AE,FE,CE,42,40,32,3C,00,19
921 AC  DATA 34,3C,00,8F,36,3C,00,2C,22,79
922 r9  DATA 00,00,3A,6A,4E,AE,FE,CE,30,3C
923 YQ  DATA 00,03,42,44,42,45,3C,3C,00,8F
924 Ay  DATA 3E,3C,00,2C,4E,B9,00,00,04,88
925 1s  DATA 4B,F9,00,00,3C,03,3A,3C,00,10
926 fc  DATA 3C,3C,00,08,3E,3C,00,08,4E,B9
927 eU  DATA 00,00,04,18,4B,F9,00,00,3B,9F
928 kV  DATA 3A,3C,00,06,3C,3C,00,30,3E,3C
929 Ba  DATA 00,25,4E,B9,00,00,04,18,30,3C
930 Wt  DATA 00,02,38,3C,00,2C,3A,3C,00,1B
931 sY  DATA 3C,3C,00,64,3E,3C,00,2A,4E,B9
932 pv  DATA 00,00,04,88,30,39,00,00,30,52
933 FP  DATA 08,00,00,07,66,00,00,36,08,39
934 XI  DATA 00,06,00,BF,E0,01,66,E8,22,79
935 Bf  DATA 00,00,3A,56,3A,29,00,0E,3C,29
936 iE  DATA 00,0C,BA,7C,00,2C,65,D4,BC,7C
937 oR  DATA 00,1B,65,CE,BA,7C,00,64,62,C8
938 D0  DATA BC,7C,00,2A,62,C2,60,00,00,0E
939 9g  DATA 30,39,00,00,30,52,08,00,00,07
940 wl  DATA 66,F4,2C,79,00,00,3A,46,20,79
941 Ps  DATA 00,00,3A,56,4E,AE,FF,B8,0C,B9
942 xR  DATA 00,00,00,00,00,00,30,8E,66,00
943 9A  DATA 00,0C,23,FC,00,00,00,01,00,00
944 3a  DATA 30,8E,20,39,00,00,3A,78,B0,B9
945 DE  DATA 00,00,30,8E,64,00,00,0C,23,F9
946 XN  DATA 00,00,3A,78,00,00,30,8E,20,39
947 hW  DATA 00,00,30,8E,33,C0,00,00,3A,76
948 Kw  DATA 90,BC,00,00,00,01,C0,FC,02,80
949 AG  DATA D0,B9,00,00,3A,4A,23,C0,00,00
950 b2  DATA 3B,10,D0,BC,00,00,00,80,23,C0
951 Bp  DATA 00,00,3B,14,D0,BC,00,00,00,80
952 sv  DATA 23,C0,00,00,3B,18,D0,BC,00,00
953 2n  DATA 00,80,23,C0,00,00,3B,1C,D0,BC
954 9P  DATA 00,00,00,80,23,C0,00,00,3B,20
955 ai  DATA 2C,79,00,00,3A,3E,20,79,00,00
956 vp  DATA 3A,62,43,F9,00,00,32,6E,30,3C
957 bS  DATA 00,20,4E,AE,FF,40,4E,B9,00,00
958 4W  DATA 2A,42,4E,B9,00,00,27,A6,60,00
959 Jy  DATA E2,8C,23,FC,00,00,00,64,00,00
960 1K  DATA 30,8E,20,3C,00,31,30,30,E1,88
961 OT  DATA 23,C0,00,00,30,96,23,C0,00,00
962 V8  DATA 30,9C,2C,79,00,00,3A,46,41,F9
963 I4  DATA 00,00,2D,C6,4E,AE,FF,34,23,C0
964 4u  DATA 00,00,3A,56,20,40,23,E8,00,32
965 ek  DATA 00,00,3A,6A,23,E8,00,32,00,00
966 yb  DATA 3A,6E,22,68,00,32,30,3C,00,01
967 V8  DATA 2C,79,00,00,3A,3E,4E,AE,FE,AA
968 QT  DATA 20,79,00,00,3A,62,30,3C,00,1F
969 rq  DATA 32,3C,00,0D,34,01,36,01,4E,AE
970 Si  DATA FE,E0,20,79,00,00,3A,62,30,3C
971 MF  DATA 00,1E,32,3C,00,03,34,01,36,01
972 Ev  DATA 4E,AE,FE,E0,22,79,00,00,3A,6A
973 kP  DATA 30,3C,00,01,4E,AE,FE,A4,22,79
974 YQ  DATA 00,00,3A,6A,42,40,42,41,34,3C
975 7X  DATA 00,8F,36,3C,00,0C,4E,AE,FE,CE
976 7L  DATA 42,40,32,3C,00,0D,34,3C,00,36
977 8C  DATA 36,3C,00,18,22,79,00,00,3A,6A
978 tE  DATA 4E,AE,FE,CE,30,3C,00,5A,32,3C
979 G6  DATA 00,0D,34,3C,00,8F,36,3C,00,18
980 Fg  DATA 22,79,00,00,3A,6A,4E,AE,FE,CE
981 Ji  DATA 42,40,32,3C,00,19,34,3C,00,8F
982 fq  DATA 36,3C,00,2C,22,79,00,00,3A,6A
983 mW  DATA 4E,AE,FE,CE,30,3C,00,03,42,44
984 oF  DATA 42,45,3C,3C,00,8F,3E,3C,00,2C
985 28  DATA 4E,B9,00,00,04,88,4B,F9,00,00
986 RO  DATA 3C,40,3A,3C,00,0F,3C,3C,00,0C
987 Bp  DATA 3E,3C,00,08,4E,B9,00,00,04,18
988 Kx  DATA 4B,F9,00,00,3B,9F,3A,3C,00,06
989 cE  DATA 3C,3C,00,30,3E,3C,00,25,4E,B9
990 CJ  DATA 00,00,04,18,30,3C,00,02,38,3C
991 CL  DATA 00,2C,3A,3C,00,1B,3C,3C,00,64
992 Cy  DATA 3E,3C,00,2A,4E,B9,00,00,04,88
993 1Y  DATA 30,39,00,00,30,52,08,00,00,07
994 Vv  DATA 66,00,00,36,08,39,00,06,00,BF
995 OT  DATA E0,01,66,E8,22,79,00,00,3A,56
996 QA  DATA 3A,29,00,0E,3C,29,00,0C,BA,7C
997 T9  DATA 00,2C,65,D4,BC,7C,00,1B,65,CE
998 WG  DATA BA,7C,00,64,62,C8,BC,7C,00,2A
999 Pe  DATA 62,C2,60,00,00,0E,30,39,00,00
1000 37 DATA 30,52,08,00,00,07,66,F4,2C,79
1001 PN DATA 00,00,3A,46,20,79,00,00,3A,56
1002 kk DATA 4E,AE,FF,B8,0C,B9,00,00,00,00
1003 7K DATA 00,00,30,8E,66,00,00,0C,23,FC
1004 er DATA 00,00,00,01,00,00,30,8E,23,F9
1005 XI DATA 00,00,30,8E,00,00,3A,78,2C,79
1006 cq DATA 00,00,3A,3E,20,79,00,00,3A,62
1007 TZ DATA 43,F9,00,00,32,6E,30,3C,00,20
1008 1Y DATA 4E,AE,FF,40,23,FC,00,00,00,01
1009 jA DATA 00,00,30,8E,20,3C,00,30,30,31
1010 9k DATA E1,88,23,C0,00,00,30,96,23,C0
1011 Ge DATA 00,00,30,9C,4E,75,08,39,00,05
1012 mU DATA 00,DF,F0,1F,66,00,00,06,60,00
1013 jb DATA 00,AC,48,E7,FF,FE,0C,B9,00,00
1014 Es DATA 00,00,00,00,3A,8A,67,00,00,10
1015 fC DATA 04,B9,00,00,00,01,00,00,3A,8A
1016 K4 DATA 60,00,00,88,20,39,00,00,3A,8E
1017 R3 DATA 90,BC,00,00,00,01,C0,FC,02,80
1018 HN DATA D0,B9,00,00,3A,4A,23,C0,00,00
1019 i9 DATA 3B,10,D0,BC,00,00,00,80,23,C0
1020 Iw DATA 00,00,3B,14,D0,BC,00,00,00,80
1021 z2 DATA 23,C0,00,00,3B,18,D0,BC,00,00
1022 9u DATA 00,80,23,C0,00,00,3B,1C,D0,BC
1023 GW DATA 00,00,00,80,23,C0,00,00,3B,20
1024 Yf DATA 33,FC,00,01,00,00,3A,96,23,F9
1025 Ce DATA 00,00,3A,92,00,00,3A,8A,20,39
1026 Ub DATA 00,00,3A,8E,B0,B9,00,00,31,46
1027 g4 DATA 66,00,00,10,23,F9,00,00,30,EA
1028 yJ DATA 00,00,3A,8E,60,00,00,0C,06,B9
1029 Kk DATA 00,00,00,01,00,00,3A,8E,4C,DF
1030 1z DATA 7F,FF,4E,F9,00,00,00,00,2C,79
1031 ca DATA 00,00,3A,3E,42,80,42,81,42,82
1032 oK DATA 42,83,30,39,00,00,3A,72,D0,7C
1033 Qk DATA 00,10,32,10,34,10,36,10,C2,7C
1034 HK DATA 0F,00,E0,59,C4,7C,00,F0,E8,5A
1035 tI DATA C6,7C,00,0F,20,79,00,00,3A,62
1036 3d DATA 4E,AE,FE,E0,4E,75,2C,79,00,00
1037 sL DATA 3A,3E,41,F9,00,00,3A,98,22,79
1038 YN DATA 00,00,3A,66,42,80,42,81,34,3C
1039 9j DATA 01,00,36,3C,00,08,38,3C,00,20
1040 UT DATA 3A,3C,00,20,3C,3C,00,C0,4E,AE
1041 jq DATA FD,D8,4E,B9,00,00,28,C0,22,79
1042 1R DATA 00,00,3A,66,30,3C,00,02,4E,AE
1043 Wv DATA FE,AA,42,86,22,79,00,00,3A,66
1044 V7 DATA 33,46,00,24,33,7C,00,00,00,26
1045 87 DATA 30,06,32,3C,00,E0,4E,AE,FF,0A
1046 cG DATA 22,79,00,00,3A,66,33,7C,00,00
1047 gW DATA 00,24,33,46,00,26,30,3C,00,E0
1048 9M DATA 32,06,4E,AE,FF,0A,DC,7C,00,07
1049 w8 DATA BC,7C,00,E7,66,C2,22,79,00,00
1050 yH DATA 3A,66,30,3C,00,04,4E,AE,FE,AA
1051 9u DATA 22,79,00,00,3A,66,33,7C,00,70
1052 hz DATA 00,24,33,7C,00,01,00,26,30,3C
1053 Bx DATA 00,70,32,3C,00,DF,4E,AE,FF,0A
1054 pU DATA 22,79,00,00,3A,66,33,7C,00,01
1055 vJ DATA 00,24,33,7C,00,70,00,26,30,3C
1056 vv DATA 00,DF,32,3C,00,70,4E,AE,FF,0A
1057 1B DATA 2C,78,00,04,4E,AE,FF,7C,2C,3C
1058 YC DATA 00,00,00,1F,2E,3C,00,00,00,1F
1059 r3 DATA 20,07,22,06,4E,B9,00,00,2A,94
1060 Qy DATA 20,07,22,06,C0,FC,00,07,C2,FC
1061 sV DATA 00,07,D0,BC,00,00,00,01,D2,BC
1062 oz DATA 00,00,00,01,4E,B9,00,00,29,30
1063 c5 DATA 51,CF,FF,D6,51,CE,FF,CC,2C,78
1064 0A DATA 00,04,4E,AE,FF,76,4E,75,4E,AE
1065 G4 DATA FE,38,2A,7C,00,DF,F0,00,20,79
1066 g5 DATA 00,00,3A,5A,22,28,00,08,4E,B9
1067 Uq DATA 00,00,28,FC,22,28,00,0C,4E,B9
1068 KO DATA 00,00,28,FC,22,28,00,10,4E,B9
1069 bj DATA 00,00,28,FC,22,28,00,14,4E,B9
1070 D6 DATA 00,00,28,FC,4E,AE,FE,32,08,2D
1071 9r DATA 00,0E,00,02,66,F8,2B,41,00,54
1072 LH DATA 3B,7C,00,0C,00,66,3B,7C,FF,FF
1073 ce DATA 00,44,3B,7C,FF,FF,00,46,42,6D
1074 ot DATA 00,42,42,6D,00,74,3B,7C,01,F0
1075 nJ DATA 00,40,3B,7C,38,0E,00,58,4E,75
1076 sT DATA 36,00,C6,BC,00,00,00,0F,24,3C
1077 Ou DATA FC,00,00,00,E6,AA,C2,FC,00,28
1078 aZ DATA E6,48,08,80,00,00,D2,80,76,28
1079 Yc DATA 20,79,00,00,3A,5A,08,05,00,00
1080 Le DATA 67,08,22,68,00,08,D3,C1,61,2C
1081 NE DATA 08,05,00,01,67,08,22,68,00,0C
1082 Tp DATA D3,C1,61,1E,08,05,00,02,67,08
1083 TG DATA 22,68,00,10,D3,C1,61,10,08,05
1084 I9 DATA 00,03,67,08,22,68,00,14,D3,C1
1085 9h DATA 61,02,4E,75,70,05,85,91,D3,C3
1086 uQ DATA 51,C8,FF,FA,4E,75,0C,B9,00,00
1087 8d DATA 00,01,00,00,3A,92,66,00,00,06
1088 M9 DATA 60,00,00,6C,04,B9,00,00,00,01
1089 w2 DATA 00,00,3A,92,41,F9,00,00,3C,3D
1090 el DATA 22,39,00,00,3A,92,82,FC,00,64
1091 hZ DATA 61,00,00,C0,82,FC,00,0A,61,00
1092 LK DATA 00,B8,61,00,00,B4,60,00,00,3E
1093 SQ DATA 0C,B9,00,00,00,C8,00,00,3A,92
1094 7d DATA 66,00,00,06,60,00,00,2C,06,B9
1095 iN DATA 00,00,00,01,00,00,3A,92,41,F9
1096 Oe DATA 00,00,3C,3D,22,39,00,00,3A,92
1097 1f DATA 82,FC,00,64,61,00,00,80,82,FC
1098 gP DATA 00,0A,61,00,00,78,61,00,00,74
1099 Lk DATA 4B,F9,00,00,3C,3D,2A,3C,00,00
1100 LZ DATA 00,03,2C,3C,00,00,00,63,2E,3C
1101 h0 DATA 00,00,00,5D,4E,B9,00,00,04,18
1102 ai DATA 08,39,00,06,00,BF,E0,01,67,F6
1103 nZ DATA 60,00,F4,BC,41,F9,00,00,3B,74
1104 2p DATA 42,81,32,39,00,00,3A,76,82,FC
1105 ky DATA 00,64,61,00,00,32,82,FC,00,0A
1106 tB DATA 61,00,00,2A,61,00,00,26,4B,F9
1107 Nm DATA 00,00,3B,74,3A,3C,00,03,3C,3C
1108 Qg DATA 01,03,3E,3C,00,C2,23,F9,00,00
1109 Se DATA 3A,66,00,00,3A,6E,4E,B9,00,00
1110 UQ DATA 04,18,4E,75,D2,7C,00,30,10,C1
1111 it DATA 42,41,48,41,4E,75,42,85,20,79
1112 a3 DATA 00,00,3B,1C,4E,B9,00,00,2A,CE
1113 g0 DATA E3,5D,20,79,00,00,3B,18,4E,B9
1114 UF DATA 00,00,2A,CE,E3,5D,20,79,00,00
1115 9Z DATA 3B,14,4E,B9,00,00,2A,CE,E3,5D
1116 zJ DATA 20,79,00,00,3B,10,4E,B9,00,00
1117 0q DATA 2A,CE,4E,75,26,00,28,01,86,FC
1118 68 DATA 00,08,C8,FC,00,04,D8,43,48,43
1119 ee DATA 14,3C,00,07,94,03,05,30,40,00
1120 DZ DATA 67,00,00,06,08,C5,00,00,4E,75
1121 zh DATA 20,79,00,00,3A,4A,20,39,00,00
1122 80 DATA 3A,78,90,BC,00,00,00,01,D1,FC
1123 OT DATA 00,00,02,00,22,3C,00,00,00,1F
1124 ON DATA 20,FC,FF,FF,FF,FF,51,C9,FF,F8
1125 wS DATA 51,C8,FF,E8,4E,75,20,79,00,00
1126 hV DATA 3A,4A,22,48,24,48,26,48,28,48
```

```
1127 Em  DATA D3,FC,00,00,00,80,D5,FC,00,00
1128 4F  DATA 01,00,D7,FC,00,00,01,80,D9,FC
1129 wm  DATA 00,00,02,00,20,39,00,00,3A,78
1130 aK  DATA 90,BC,00,00,00,01,22,3C,00,00
1131 xm  DATA 00,1F,42,87,24,18,8E,82,24,19
1132 pY  DATA 8E,82,24,1A,8E,82,24,1B,8E,82
1133 4b  DATA 28,C7,51,C9,FF,EA,D1,FC,00,00
1134 XP  DATA 02,00,D3,FC,00,00,02,00,D5,FC
1135 zP  DATA 00,00,02,00,D7,FC,00,00,02,00
1136 Lx  DATA D9,FC,00,00,02,00,51,C8,FF,C2
1137 DW  DATA 4E,75,4E,B9,00,00,01,10,4E,B9
1138 Om  DATA 00,00,00,06,4E,B9,00,00,00,66
1139 Wp  DATA 4E,B9,00,00,2C,9C,4E,B9,00,00
1140 52  DATA 01,54,4E,B9,00,00,24,A0,2C,78
1141 Ha  DATA 00,04,22,3C,00,01,00,03,20,39
1142 to  DATA 00,00,3A,78,C0,FC,02,80,4E,AE
1143 sG  DATA FF,3A,23,C0,00,00,3A,4A,67,DA
1144 aV  DATA 23,C0,00,00,3B,10,D0,BC,00,00
1145 Ag  DATA 00,80,23,C0,00,00,3B,14,D0,BC
1146 pC  DATA 00,00,00,80,23,C0,00,00,3B,18
1147 4z  DATA D0,BC,00,00,00,80,23,C0,00,00
1148 dq  DATA 3B,1C,D0,BC,00,00,00,80,23,C0
1149 AP  DATA 00,00,3B,20,33,FC,00,01,00,00
1150 Ni  DATA 3A,76,4E,B9,00,00,2A,F2,08,39
1151 EJ  DATA 00,06,00,BF,E0,01,67,00,00,10
1152 MM  DATA 08,39,00,0A,00,DF,F0,16,67,00
1153 Gu  DATA 00,BE,66,E6,22,79,00,00,3A,52
1154 RP  DATA 42,85,42,86,3A,29,00,0E,3C,29
1155 D5  DATA 00,0C,BA,7C,00,E0,6C,00,DA,86
1156 9Z  DATA BC,7C,00,E0,6C,00,D8,84,61,00
1157 nq  DATA D9,E0,60,BE,2C,78,00,04,20,39
1158 j7  DATA 00,00,3A,78,C0,FC,02,80,22,79
1159 ez  DATA 00,00,3A,4A,4E,AE,FF,2E,2C,78
1160 J6  DATA 00,04,20,3C,00,00,08,00,22,79
1161 x0  DATA 00,00,3A,5E,4E,AE,FF,2E,4E,B9
1162 G2  DATA 00,00,00,F0,4E,B9,00,00,00,42
1163 pk  DATA 4E,F9,00,00,01,3C,2C,78,00,04
1164 Nt  DATA 20,3C,00,00,08,00,22,3C,00,01
1165 RK  DATA 00,03,4E,AE,FF,3A,23,C0,00,00
1166 QH  DATA 3A,5E,22,40,20,7C,00,00,36,3E
1167 3N  DATA 20,3C,00,00,01,FF,22,D8,51,C8
1168 5p  DATA FF,FC,22,39,00,00,3A,5E,41,F9
1169 7f  DATA 00,00,34,36,D1,FC,00,00,00,0A
1170 kg  DATA 20,3C,00,00,00,19,D3,90,D1,FC
1171 nF  DATA 00,00,00,14,51,C8,FF,F6,4E,75
1172 aL  DATA 22,79,00,00,3A,52,3A,29,00,0E
1173 zM  DATA 3C,29,00,0C,BA,7C,00,E0,64,00
1174 gN  DATA 00,2E,BC,7C,00,E0,64,00,00,26
1175 5t  DATA 33,F9,00,00,3A,72,00,00,3A,80
1176 n0  DATA 33,FC,00,00,00,00,3A,72,61,00
1177 x5  DATA D9,18,33,F9,00,00,3A,80,00,00
1178 LN  DATA 3A,72,60,00,FE,EC,BA,7C,01,3F
1179 hH  DATA 66,00,FE,E4,BC,7C,00,00,66,00
1180 Yh  DATA FE,DC,60,00,FF,1A,00,00,00,00
1181 xR  DATA 01,40,01,00,00,05,00,01,00,02
1182 C2  DATA 00,0F,00,00,00,00,00,00,00,00
1183 5L  DATA 00,00,00,00,00,00,00,00,00,00
1184 hk  DATA 00,00,01,40,01,00,00,01,00,00
1185 7f  DATA 00,00,00,01,18,80,00,00,00,00
1186 80  DATA 00,00,00,00,00,00,00,00,00,00
1187 Tk  DATA 00,00,00,00,00,00,01,40,01,00
1188 oY  DATA 01,40,01,00,00,0F,00,01,00,01
1189 sB  DATA 00,DF,00,DF,02,03,00,00,00,00
1190 rC  DATA 00,01,18,00,00,00,00,00,00,00
1191 DT  DATA 00,00,00,00,00,00,00,00,00,00
1192 Og  DATA 00,00,00,00,00,DF,00,DF,00,DF
1193 Qi  DATA 00,DF,00,0F,00,58,00,5E,00,90
1194 fl  DATA 00,2D,02,03,00,00,00,00,00,01
1195 Jv  DATA 18,00,00,00,30,46,00,00,00,00
1196 IY  DATA 00,00,00,00,00,00,00,00,00,00
1197 Ov  DATA 00,00,00,90,00,2D,00,90,00,2D
1198 yg  DATA 00,0F,00,00,2E,76,00,60,00,57
1199 DL  DATA 00,20,00,0A,00,00,08,01,00,04
1200 4b  DATA 00,00,2E,22,00,00,00,00,00,00
1201 zc  DATA 00,00,00,00,00,00,00,00,2E,46
1202 Ug  DATA 00,01,00,00,00,00,00,00,00,00
1203 BK  DATA 03,00,00,05,00,00,2E,32,00,00
1204 gd  DATA 00,00,FF,FE,FF,FE,00,20,FF,FE
1205 Lx  DATA 00,20,00,09,FF,FE,00,09,FF,FE
1206 CE  DATA FF,FE,00,00,2E,6A,00,00,2E,70
1207 bv  DATA 00,00,00,04,00,00,00,00,00,00
1208 Uk  DATA 00,00,00,00,00,00,00,00,00,00
1209 z6  DATA 00,00,00,00,00,00,00,00,30,30
1210 Ih  DATA 31,00,00,00,30,30,31,00,00,00
1211 Bv  DATA 00,00,00,00,00,27,00,75,00,90
1212 Vk  DATA 00,0A,00,00,00,01,00,04,00,00
1213 Dz  DATA 2E,A2,00,00,00,00,00,00,00,00
1214 Fc  DATA 00,00,00,00,00,00,2E,C6,00,02
1215 h0  DATA 00,00,00,00,00,00,00,00,03,00
1216 bE  DATA 00,05,00,00,2E,B2,00,00,00,00
1217 hD  DATA FF,FE,FF,FE,00,90,FF,FE,00,90
1218 WE  DATA 00,09,FF,FE,00,09,FF,FE,FF,FE
1219 IX  DATA 00,00,2E,EA,00,00,2F,14,00,00
1220 c0  DATA 00,28,00,00,00,00,00,00,00,00
1221 hx  DATA 00,00,00,00,00,00,00,00,00,00
1222 Vb  DATA 00,00,00,00,00,00,4F,62,6A,65
1223 Zh  DATA 6B,74,65,2E,41,4F,45,00,00,00
1224 k0  DATA 00,00,00,00,00,00,00,00,00,00
1225 11  DATA 00,00,00,00,00,00,00,00,00,00
1226 Qu  DATA 00,00,00,00,00,00,00,00,4F,62
1227 Ug  DATA 6A,65,6B,74,65,2E,41,4F,45,00
1228 o4  DATA 00,00,00,00,00,00,00,00,00,00
1229 p5  DATA 00,00,00,00,00,00,00,00,00,00
1230 q6  DATA 00,00,00,00,00,00,00,00,00,00
1231 6q  DATA 00,00,2F,9A,00,38,00,57,00,20
1232 Tq  DATA 00,0A,00,00,08,01,00,04,00,00
1233 Kr  DATA 2E,22,00,00,00,00,00,00,00,00
1234 fv  DATA 00,00,00,00,00,00,2F,6A,00,01
1235 xI  DATA 00,00,00,00,00,00,2F,8E,00,00
1236 Ny  DATA 2F,94,00,00,00,04,00,00,00,00
1237 xD  DATA 00,00,00,00,00,00,00,00,00,00
1238 yE  DATA 00,00,00,00,00,00,00,00,00,00
1239 Fa  DATA 30,30,31,00,00,00,30,30,31,00
1240 hF  DATA 00,00,00,00,2F,F6,00,88,00,57
1241 t1  DATA 00,20,00,0A,00,00,08,01,00,04
1242 kH  DATA 00,00,2E,22,00,00,00,00,00,00
1243 fY  DATA 00,00,00,00,00,00,00,00,2F,C6
1244 5v  DATA 00,04,00,00,00,00,00,00,2F,EA
1245 zH  DATA 00,00,2F,F0,00,00,00,04,00,00
1246 6M  DATA 00,00,00,00,00,00,00,00,00,00
1247 7N  DATA 00,00,00,00,00,00,00,00,00,00
1248 KM  DATA 00,00,30,30,31,00,00,00,30,30
1249 Qm  DATA 31,00,00,00,00,00,00,00,00,20
1250 zk  DATA 00,75,00,90,00,0A,00,00,00,01
1251 2i  DATA 00,04,00,00,2E,A2,00,00,00,00
1252 CS  DATA 00,00,00,00,00,00,00,00,00,00
1253 p7  DATA 30,22,00,05,00,00,00,00,00,00
1254 Y3  DATA 2E,EA,00,00,2F,14,00,00,00,28
1255 FV  DATA 00,00,00,00,00,00,00,00,00,00
1256 GW  DATA 00,00,00,00,00,00,00,00,00,00
1257 Ce  DATA 00,00,00,00,00,00,00,00,00,39
1258 mu  DATA 00,0F,00,20,00,0A,00,00,08,01
1259 g7  DATA 00,04,00,00,2E,22,00,00,00,00
1260 Ka  DATA 00,00,00,00,00,00,00,00,00,00
1261 EX  DATA 30,72,00,01,00,00,00,00,00,00
1262 U7  DATA 30,96,00,00,30,9C,00,00,00,04
1263 Nd  DATA 00,00,00,00,00,00,00,00,00,00
1264 6W  DATA 00,00,00,00,00,00,00,00,00,64
1265 1R  DATA 00,00,00,00,31,30,30,00,00,00
1266 MH  DATA 31,30,30,00,00,00,00,00,30,FE
1267 Ex  DATA 00,38,00,61,00,20,00,0A,00,00
1268 Uc  DATA 08,01,00,04,00,00,2E,22,00,00
1269 Tj  DATA 00,00,00,00,00,00,00,00,00,00
1270 Cd  DATA 00,00,30,CE,00,01,00,00,00,00
1271 f7  DATA 00,00,30,F2,00,00,30,F8,00,00
1272 uu  DATA 00,04,00,00,00,00,00,00,00,00
1273 Xn  DATA 00,00,00,00,00,00,00,00,00,00
1274 MT  DATA 00,01,00,00,00,00,30,30,31,00
1275 HS  DATA 00,00,30,30,31,00,00,00,00,00
1276 Kc  DATA 00,00,00,88,00,61,00,20,00,0A
1277 VF  DATA 00,00,08,01,00,04,00,00,2E,22
1278 cs  DATA 00,00,00,00,00,00,00,00,00,00
1279 3k  DATA 00,00,00,00,31,2A,00,04,00,00
1280 2j  DATA 00,00,00,00,31,4E,00,00,31,54
1281 n7  DATA 00,00,00,04,00,00,00,00,00,00
1282 gw  DATA 00,00,00,00,00,00,00,00,00,00
1283 FO  DATA 00,00,00,02,00,00,00,00,30,30
1284 dz  DATA 32,00,00,00,30,30,32,00,00,00
1285 Bq  DATA 00,00,31,B6,00,38,00,57,00,20
1286 Li  DATA 00,0A,00,00,08,01,00,04,00,00
1287 CJ  DATA 2E,22,00,00,00,00,00,00,00,00
1288 yS  DATA 00,00,00,00,00,00,31,86,00,01
1289 3q  DATA 00,00,00,00,00,00,31,AA,00,00
1290 bH  DATA 31,B0,00,00,00,04,00,00,00,00
1291 p5  DATA 00,00,00,00,00,00,00,00,00,00
1292 tA  DATA 00,00,00,00,00,01,00,00,00,00
1293 7S  DATA 30,30,31,00,00,00,30,30,31,00
1294 9z  DATA 00,00,00,00,32,12,00,88,00,57
1295 1t  DATA 00,20,00,0A,00,00,08,01,00,04
1296 c9  DATA 00,00,2E,22,00,00,00,00,00,00
1297 1D  DATA 00,00,00,00,00,00,00,00,31,E2
1298 A6  DATA 00,04,00,00,00,00,00,00,32,06
1299 6o  DATA 00,00,32,0C,00,00,00,04,00,00
1300 yE  DATA 00,00,00,00,00,00,00,00,00,00
1301 7P  DATA 00,00,00,00,00,00,00,02,00,00
1302 HK  DATA 00,00,30,30,32,00,00,00,30,30
1303 Cx  DATA 32,00,00,00,00,00,00,00,00,7C
1304 c6  DATA 00,6B,00,20,00,0A,00,00,08,01
1305 Qr  DATA 00,04,00,00,2E,22,00,00,00,00
1306 4K  DATA 00,00,00,00,00,00,00,00,00,00
1307 gm  DATA 32,3E,00,04,00,00,00,00,00,00
1308 Eu  DATA 32,62,00,00,32,68,00,00,00,04
1309 7N  DATA 00,00,00,00,00,00,00,00,00,00
1310 DU  DATA 00,00,00,00,00,00,00,00,00,01
1311 m7  DATA 00,00,00,00,30,30,31,00,00,00
1312 bG  DATA 30,30,31,00,00,00,00,00,05,55
1313 5U  DATA 09,99,0B,BB,0E,EE,00,00,00,00
1314 CS  DATA 00,00,00,00,00,00,00,00,00,00
1315 DT  DATA 00,00,00,00,00,00,00,00,00,00
1316 5C  DATA 00,07,00,0F,07,7F,07,07,0F,0F
1317 Ab  DATA 0F,7F,03,33,0A,AA,0F,FF,07,00
1318 9p  DATA 0F,00,0F,77,00,70,00,F0,07,F7
1319 K2  DATA 00,02,00,E8,00,0C,00,F8,00,10
1320 uV  DATA 00,E8,00,1A,00,F8,00,1E,00,E8
1321 pG  DATA 00,28,00,F8,00,2C,00,E8,00,36
1322 IY  DATA 00,F8,00,3A,00,E8,00,44,00,F8
1323 Nj  DATA 00,48,00,E8,00,52,00,F8,00,56
1324 3r  DATA 00,E8,00,60,00,F8,00,64,00,E8
1325 7r  DATA 00,6E,00,F8,00,72,00,E8,00,7C
1326 KG  DATA 00,F8,00,80,00,E8,00,8A,00,F8
1327 kl  DATA 00,8E,00,E8,00,98,00,F8,00,9C
1328 kK  DATA 00,E8,00,A6,00,F8,00,AA,00,E8
1329 Ym  DATA 00,B4,00,F8,00,B8,00,E8,00,C2
1330 8N  DATA 00,F8,00,C6,00,E8,00,D0,00,F8
1331 lA  DATA 00,D4,00,E8,00,DE,00,F8,01,09
1332 2L  DATA 00,50,01,18,00,5F,01,09,00,2E
1333 jX  DATA 01,18,00,3D,01,1A,00,3F,01,29
1334 vS  DATA 00,4E,00,F8,00,3F,01,07,00,4E
1335 Xy  DATA 01,09,00,3F,01,18,00,4E,01,00
1336 wd  DATA 00,66,01,0F,00,75,01,11,00,66
1337 Iq  DATA 01,20,00,75,01,00,00,77,01,0F
1338 19  DATA 00,86,01,11,00,77,01,20,00,86
1339 tS  DATA 01,00,00,88,01,0F,00,97,01,11
1340 KZ  DATA 00,88,01,20,00,97,01,00,00,99
1341 3K  DATA 01,0F,00,A8,01,11,00,99,01,20
1342 d1  DATA 00,A8,00,F8,00,BC,00,FF,00,C3
1343 hJ  DATA 01,20,00,BC,01,27,00,C3,00,FB
1344 w2  DATA 00,E2,01,02,00,E9,00,FB,00,F6
1345 4r  DATA 01,02,00,FD,01,0B,00,E2,01,12
1346 Ef  DATA 00,E9,01,0B,00,F6,01,12,00,FD
1347 5m  DATA 01,1B,00,E2,01,22,00,E9,01,1B
1348 xK  DATA 00,F6,01,22,00,FD,01,01,00,BC
1349 Oq  DATA 01,25,00,C3,00,00,07,42,00,00
1350 kS  DATA 07,A0,00,00,07,FA,00,00,08,48
1351 db  DATA 00,00,08,96,00,00,0A,28,00,00
1352 mT  DATA 14,6E,00,00,14,A4,00,00,15,38
1353 tS  DATA 00,00,15,84,00,00,16,02,00,00
1354 x0  DATA 19,A0,00,00,1A,F4,00,00,20,46
1355 v7  DATA 00,00,20,A2,00,00,21,00,00,00
1356 bG  DATA 21,3C,00,00,21,78,00,00,21,B4
1357 96  DATA 00,00,21,F0,00,00,22,28,00,00
1358 f7  DATA 22,60,00,00,00,00,00,10,00,10
1359 PL  DATA 00,02,00,00,00,00,03,00,00,00
1360 Au  DATA 34,4A,00,00,FF,DE,00,10,00,10
1361 ZZ  DATA 00,02,00,00,00,40,03,00,00,00
1362 sM  DATA 34,5E,00,11,FF,EF,00,10,00,10
1363 jn  DATA 00,02,00,00,00,80,03,00,00,00
1364 Kp  DATA 34,72,FF,EF,FF,EF,00,10,00,10
1365 7M  DATA 00,02,00,00,00,C0,03,00,00,00
1366 b9  DATA 34,86,00,00,FF,EF,00,10,00,10
1367 cZ  DATA 00,02,00,00,01,00,03,00,00,00
1368 BT  DATA 00,00,00,00,00,00,00,10,00,10
1369 hh  DATA 00,02,00,00,00,40,03,00,00,00
1370 lS  DATA 34,AE,00,00,FF,DE,00,10,00,10
1371 bX  DATA 00,02,00,00,00,00,03,00,00,00
1372 Q7  DATA 34,C2,FF,EF,FF,EF,00,10,00,10
1373 tx  DATA 00,02,00,00,00,80,03,00,00,00
1374 pW  DATA 34,D6,00,11,FF,EF,00,10,00,10
1375 HW  DATA 00,02,00,00,00,C0,03,00,00,00
1376 sv  DATA 34,EA,00,00,FF,EF,00,10,00,10
```

Listing 1. (Fortsetzung)

```
1377 uv  DATA 00,02,00,00,01,40,03,00,00,00
1378 Ld  DATA 00,00,00,00,00,00,00,10,00,10
1379 49  DATA 00,02,00,00,01,80,03,00,00,00
1380 7N  DATA 35,12,00,11,00,00,00,10,00,10
1381 Si  DATA 00,02,00,00,01,C0,03,00,00,00
1382 Zf  DATA 35,26,00,00,00,11,00,10,00,10
1383 xv  DATA 00,02,00,00,02,00,03,00,00,00
1384 sE  DATA 35,3A,00,11,00,11,00,10,00,10
1385 79  DATA 00,02,00,00,02,40,03,00,00,00
1386 KW  DATA 35,4E,00,00,00,22,00,10,00,10
1387 HN  DATA 00,02,00,00,02,80,03,00,00,00
1388 oD  DATA 35,62,00,11,00,22,00,10,00,10
1389 fw  DATA 00,02,00,00,02,C0,03,00,00,00
1390 GV  DATA 35,76,00,00,00,33,00,10,00,10
1391 A9  DATA 00,02,00,00,03,00,03,00,00,00
1392 Z4  DATA 35,8A,00,11,00,33,00,10,00,10
1393 KN  DATA 00,02,00,00,03,40,03,00,00,00
1394 eA  DATA 00,00,00,00,00,00,00,08,00,08
1395 Ub  DATA 00,02,00,00,03,80,03,00,00,00
1396 7v  DATA 35,B2,00,28,00,00,00,08,00,08
1397 o4  DATA 00,02,00,00,03,A0,03,00,00,00
1398 iE  DATA 00,00,00,00,00,00,00,08,00,08
1399 uC  DATA 00,02,00,00,03,C0,03,00,00,00
1400 ZW  DATA 35,DA,00,00,00,14,00,08,00,08
1401 OK  DATA 00,02,00,00,03,E0,03,00,00,00
1402 wV  DATA 35,EE,00,10,00,00,00,08,00,08
1403 yG  DATA 00,02,00,00,03,C0,03,00,00,00
1404 1G  DATA 36,02,00,10,00,14,00,08,00,08
1405 40  DATA 00,02,00,00,03,E0,03,00,00,00
1406 5F  DATA 36,16,00,20,00,00,00,08,00,08
1407 2K  DATA 00,02,00,00,03,C0,03,00,00,00
1408 U2  DATA 36,2A,00,20,00,14,00,08,00,08
1409 8S  DATA 00,02,00,00,03,E0,03,00,00,00
1410 TO  DATA 00,00,00,00,7B,FE,7B,FE,7B,FE
1411 hx  DATA 7B,FE,7B,FE,7B,FE,7B,FE,7B,FE
1412 EQ  DATA 5F,FE,6F,FE,77,FE,7B,FE,7D,FE
1413 G5  DATA 7E,FE,00,00,00,01,00,21,00,21
1414 m2  DATA 00,21,00,21,00,21,00,21,00,21
1415 0X  DATA 00,21,00,05,00,09,00,11,00,21
1416 bt  DATA 00,41,00,81,7F,FF,00,00,7F,7E
1417 FF  DATA 7F,BE,7F,DE,7F,EE,7F,F6,7F,FA
1418 Si  DATA 7F,DE,7F,DE,7F,DE,7F,DE,7F,DE
1419 AW  DATA 7F,DE,7F,DE,7F,DE,00,00,00,01
1420 b0  DATA 01,01,02,01,04,01,08,01,10,01
1421 Pd  DATA 20,01,04,01,04,01,04,01,04,01
1422 vi  DATA 04,01,04,01,04,01,04,01,7F,FF
1423 5s  DATA 00,00,7F,FE,7F,FE,7F,FE,7F,FE
1424 ku  DATA 7F,FE,7F,FE,7F,FE,7F,FC,7F,FA
1425 u0  DATA 00,76,7F,EE,7F,DE,7F,BE,7F,FE
1426 Yy  DATA 00,00,00,01,00,01,00,41,00,21
1427 Xz  DATA 00,11,7F,89,00,05,00,03,00,01
1428 Mc  DATA 00,01,00,01,00,01,00,01,00,01
1429 Iq  DATA 00,01,7F,FF,00,00,7F,FE,7D,FE
1430 iA  DATA 7B,FE,77,FE,6E,00,5F,FE,3F,FE
1431 JZ  DATA 7F,FE,7F,FE,7F,FE,7F,FE,7F,FE
1432 u3  DATA 7F,FE,7F,FE,00,00,00,01,00,01
1433 Rh  DATA 00,01,00,01,00,01,00,01,00,01
1434 dl  DATA 00,01,40,01,20,01,11,FF,08,01
1435 pY  DATA 04,01,02,01,00,01,7F,FF,00,00
1436 MX  DATA 7F,BE,7F,BE,7F,DE,7F,BE,6F,7E
1437 1L  DATA 17,FE,FB,FE,7F,DE,7F,E8,7E,F6
1438 Ds  DATA 7D,FE,7B,FE,7D,FE,7D,FE,00,00
1439 HN  DATA 00,01,02,01,02,01,04,01,02,01
1440 xG  DATA 01,09,00,17,00,21,04,01,68,01
1441 9S  DATA 10,81,00,41,00,21,00,41,00,41
1442 Dg  DATA 7F,FF,00,00,7F,7E,7F,BE,7F,DE
1443 bb  DATA 7F,BE,6F,BE,53,FE,3F,FE,7F,FC
1444 f3  DATA 7F,CA,7D,F6,7D,FE,7B,FE,7D,FE
1445 iR  DATA 7E,FE,00,00,00,01,01,01,02,01
1446 5k  DATA 04,01,02,01,02,09,00,35,00,03
1447 dT  DATA 40,01,2C,01,10,41,00,41,00,21
1448 5c  DATA 00,41,00,81,7F,FF,00,00,7F,FE
1449 hD  DATA 47,CE,5F,F6,5F,FA,5F,FA,5F,FA
1450 gw  DATA 5F,FA,5F,FA,5F,FA,5F,FA,5F,FA
1451 LZ  DATA 5F,FA,40,02,7F,FE,00,00,00,01
1452 Jj  DATA 00,01,07,F1,07,49,07,45,07,C5
1453 3J  DATA 00,05,00,05,00,05,00,05,00,05
1454 Gx  DATA 00,05,00,05,1F,FD,00,01,7F,FF
1455 wb  DATA 00,00,7F,FE,03,86,3F,BE,3B,B6
1456 7q  DATA 3B,B6,3B,B6,3B,B6,3B,BE,3B,AE
1457 Qo  DATA 3B,AE,22,36,7F,F6,7F,FE,7F,FE
1458 YC  DATA 00,00,00,01,00,01,00,01,3E,39
1459 To  DATA 22,25,22,25,22,25,22,25,22,39
1460 qw  DATA 22,29,22,29,22,25,3F,E5,00,01
1461 kN  DATA 00,01,7F,FF,00,00,7E,FE,7E,FE
1462 Cj  DATA 7E,FE,76,EE,68,96,5E,FA,3E,FC
1463 V1  DATA 7E,FE,7E,FE,7E,FE,7E,FE,7E,FE
1464 JQ  DATA 7E,FE,7E,FE,00,00,00,01,00,01
1465 Xn  DATA 00,81,00,81,00,81,00,81,00,81
1466 Gp  DATA 00,81,40,83,20,85,16,E9,08,91
1467 h4  DATA 00,81,00,81,00,81,7F,FF,00,00
1468 ps  DATA 7E,FE,7D,FE,7B,FE,77,FE,7B,FE
1469 ik  DATA 7B,FE,00,02,7F,FE,7B,FE,7B,FE
1470 Ca  DATA 77,FE,7B,FE,7D,FE,7E,FE,00,00
1471 7y  DATA 00,01,00,81,00,41,00,21,00,11
1472 Lr  DATA 00,21,00,21,00,01,3F,FF,00,21
1473 7Z  DATA 00,21,00,11,00,21,00,41,00,81
1474 zp  DATA 7F,FF,00,00,7F,FE,77,FE,6F,FE
1475 fu  DATA 5F,FE,6F,FE,6F,FE,6F,FE,6F,FE
1476 3I  DATA 6F,FE,6F,FE,6F,FE,6F,FE,60,02
1477 F5  DATA 7F,FE,00,00,00,01,00,01,04,01
1478 kz  DATA 02,01,01,01,02,01,02,01,02,01
1479 r1  DATA 02,01,02,01,03,FD,00,05,00,05
1480 VU  DATA 00,01,00,01,7F,FF,00,00,7F,FE
1481 Ox  DATA 40,0E,5F,EE,5F,FE,5F,EE,5F,EE
1482 G1  DATA 5F,EE,5F,EE,5F,EE,5F,EE,48,0E
1483 cu  DATA 7F,FE,7F,FE,7F,FE,00,00,00,01
1484 Gx  DATA 00,01,00,01,00,01,0F,FD,08,05
1485 9P  DATA 08,05,08,05,08,05,08,05,08,05
1486 cY  DATA 08,05,08,05,0F,FD,00,01,7F,FF
1487 U0  DATA 00,00,7F,FE,7C,FE,79,FE,7B,FE
1488 40  DATA 7F,FE,78,3E,7D,FE,7D,FE,7D,FE
1489 G5  DATA 7D,FE,7D,FE,71,DE,7F,FE,7F,FE
1490 WA  DATA 00,00,00,01,00,01,00,01,01,C1
1491 aS  DATA 03,E1,01,C1,00,01,03,C1,01,C1
1492 Ed  DATA 01,C1,01,C1,01,C1,01,C1,07,F1
1493 g1  DATA 00,01,7F,FF,00,00,7F,FE,63,FE
1494 FH  DATA 5D,FE,5D,FE,5D,FE,63,FE,77,FE
1495 aS  DATA 78,3E,77,FE,47,FE,47,FE,77,FE
1496 Fu  DATA 77,FE,77,FE,00,00,00,01,00,01
1497 Bg  DATA 00,01,00,31,00,49,00,49,00,31
1498 1C  DATA 00,21,00,11,00,09,00,05,00,05
1499 7U  DATA 00,05,07,FD,00,01,7F,FF,00,00
1500 cS  DATA 6E,00,52,00,3E,00,7E,00,7E,00
1501 fL  DATA 7E,00,00,00,01,00,01,00,01,00
1502 qP  DATA 01,00,41,00,2D,00,11,00,7F,00
1503 yk  DATA 00,00,7E,00,7E,00,7E,00,7C,00
1504 KD  DATA 4A,00,76,00,00,00,01,00,09,00
1505 1Q  DATA 35,00,03,00,01,00,01,00,01,00
1506 1p  DATA 7F,00,00,00,76,00,7A,00,7C,00
1507 yT  DATA 7A,00,7A,00,7E,00,00,00,01,00
1508 Of  DATA 11,00,21,00,41,00,21,00,21,00
1509 uQ  DATA 01,00,7F,00,00,00,7E,00,5E,00
1510 Zn  DATA 5E,00,3E,00,5E,00,6E,00,00,00
1511 TZ  DATA 01,00,01,00,05,00,05,00,03,00
1512 RO  DATA 05,00,09,00,7F,00,00,00,00,00
1513 Pf  DATA 00,00,00,00,00,00,00,00,00,00
1514 Qg  DATA 00,00,00,00,00,00,00,00,00,00
1515 Rh  DATA 00,00,00,00,00,00,00,00,00,00
1516 Si  DATA 00,00,00,00,00,00,00,00,00,00
1517 Yp  DATA 00,00,00,00,00,00,00,00,00,01
1518 Wn  DATA 00,00,00,01,00,00,00,00,00,00
1519 V1  DATA 00,00,00,00,00,00,00,00,00,00
1520 Wm  DATA 00,00,00,00,00,00,00,00,00,00
1521 h2  DATA 00,00,00,05,00,00,00,00,00,00
1522 Yo  DATA 00,00,00,00,00,00,00,00,00,00
1523 Zp  DATA 00,00,00,00,00,00,00,00,00,00
1524 aq  DATA 00,00,00,00,00,00,00,00,00,00
1525 br  DATA 00,00,00,00,00,00,00,00,00,00
1526 cs  DATA 00,00,00,00,00,00,00,00,00,00
1527 dt  DATA 00,00,00,00,00,00,00,00,00,00
1528 eu  DATA 00,00,00,00,00,00,00,00,00,00
1529 fv  DATA 00,00,00,00,00,00,00,00,00,00
1530 gw  DATA 00,00,00,00,00,00,00,00,00,00
1531 hx  DATA 00,00,00,00,00,00,00,00,00,00
1532 2M  DATA 00,00,00,00,00,00,00,00,00,04
1533 3Q  DATA 00,20,00,05,00,00,00,00,00,00
1534 k0  DATA 00,00,00,00,00,00,00,00,00,00
1535 11  DATA 00,00,00,00,00,00,00,00,00,00
1536 CK  DATA 00,00,00,00,00,00,00,00,67,72
1537 P9  DATA 61,70,68,69,63,73,2E,6C,69,62
1538 TM  DATA 72,61,72,79,00,64,6F,73,2E,6C
1539 OP  DATA 69,62,72,61,72,79,00,69,6E,74
1540 d0  DATA 75,69,74,69,6F,6E,2E,6C,69,62
1541 29  DATA 72,61,72,79,00,4F,62,6A,65,6B
1542 FM  DATA 74,20,4E,72,2E,52,20,47,20,42
1543 Fs  DATA 30,20,30,20,30,20,30,30,31,4C
1544 E1  DATA 61,64,65,6E,53,70,65,69,63,68
1545 VC  DATA 65,72,6E,4C,61,64,65,6E,20,6E
1546 x8  DATA 61,63,68,20,4F,62,6A,65,6B,74
1547 AR  DATA 44,61,74,65,69,6E,61,6D,65,46
1548 ZS  DATA 65,72,74,69,67,53,70,65,69,63
1549 OO  DATA 68,65,72,6E,20,76,6F,6E,20,4F
1550 QV  DATA 62,6A,65,6B,74,6E,61,63,68,41
1551 YH  DATA 4D,49,47,41,20,4F,42,4A,45,43
1552 oM  DATA 54,45,44,49,54,4F,52,20,56,31
1553 aN  DATA 2E,32,77,72,69,74,74,65,6E,20
1554 32  DATA 31,39,38,38,20,62,79,4D,49,43
1555 Fk  DATA 48,41,45,4C,20,42,45,52,54,53
1556 K3  DATA 43,48,A9,20,4D,61,72,6B,74,20
1557 KC  DATA 26,20,54,65,63,68,6E,69,6B,5A
1558 N6  DATA 65,69,67,65,20,4F,62,6A,65,6B
1559 1U  DATA 74,20,4E,72,2E,41,62,62,72,75
1560 1r  DATA 63,68,41,6E,69,6D,61,74,69,6F
1561 Vi  DATA 6E,20,76,6F,6E,20,4F,62,6A,65
1562 dJ  DATA 6B,74,47,65,73,63,68,77,69,6E
1563 GZ  DATA 64,69,67,6B,65,69,74,30,30,35
1564 bb  DATA 4D,61,78,2E,20,4F,62,6A,65,6B
1565 b0  DATA 74,7A,61,68,6C,4B,6F,70,69,65
1566 qz  DATA 72,65,6E,20,76,6F,6E,20,4F,62
1567 Oa  DATA 6A,65,6B,74,41,6E,7A,61,68,6C
1568 3I  DATA 00,00,03,EC,00,00,03,56,00,00
1569 wX  DATA 00,00,00,00,00,02,00,00,00,0C
1570 51  DATA 00,00,00,18,00,00,00,1E,00,00
1571 yZ  DATA 00,2A,00,00,00,30,00,00,00,3C
1572 Nc  DATA 00,00,00,48,00,00,00,52,00,00
1573 oI  DATA 00,5C,00,00,00,68,00,00,00,72
1574 CT  DATA 00,00,00,7C,00,00,00,82,00,00
1575 Kw  DATA 00,88,00,00,00,8E,00,00,00,98
1576 7t  DATA 00,00,00,A4,00,00,00,AA,00,00
1577 Ca  DATA 00,B2,00,00,00,BA,00,00,00,C6
1578 Oh  DATA 00,00,00,CC,00,00,00,D2,00,00
1579 PI  DATA 00,D8,00,00,00,E6,00,00,00,F2
1580 62  DATA 00,00,00,F8,00,00,01,02,00,00
1581 V3  DATA 01,36,00,00,01,3E,00,00,01,48
1582 pC  DATA 00,00,01,56,00,00,01,5C,00,00
1583 Ej  DATA 01,66,00,00,01,6A,00,00,01,70
1584 14  DATA 00,00,01,7E,00,00,01,94,00,00
1585 iP  DATA 01,A2,00,00,01,AE,00,00,01,BE
1586 Ew  DATA 00,00,01,D8,00,00,01,FC,00,00
1587 1L  DATA 02,0A,00,00,02,28,00,00,02,46
1588 4N  DATA 00,00,02,54,00,00,02,6C,00,00
1589 JO  DATA 02,86,00,00,02,94,00,00,02,B6
1590 GY  DATA 00,00,02,D0,00,00,02,D6,00,00
1591 Cc  DATA 02,E4,00,00,02,FE,00,00,03,0A
1592 Ko  DATA 00,00,03,1C,00,00,03,44,00,00
1593 1Q  DATA 03,5E,00,00,03,64,00,00,03,6A
1594 OD  DATA 00,00,03,70,00,00,03,82,00,00
1595 qY  DATA 03,88,00,00,03,9A,00,00,03,A0
1596 Qp  DATA 00,00,03,B2,00,00,03,B8,00,00
1597 V7  DATA 03,CA,00,00,03,DC,00,00,03,E2
1598 g4  DATA 00,00,03,F4,00,00,03,FA,00,00
1599 qP  DATA 04,0C,00,00,04,12,00,00,04,1A
1600 BG  DATA 00,00,04,20,00,00,04,36,00,00
1601 4w  DATA 04,42,00,00,04,50,00,00,04,6C
1602 CU  DATA 00,00,04,7A,00,00,04,8A,00,00
1603 Vo  DATA 04,90,00,00,04,9A,00,00,04,B0
1604 5w  DATA 00,00,04,BE,00,00,04,CC,00,00
1605 mM  DATA 04,DE,00,00,05,04,00,00,05,22
1606 Qk  DATA 00,00,05,26,00,00,05,2C,00,00
1607 pl  DATA 05,32,00,00,05,3E,00,00,05,62
1608 B7  DATA 00,00,05,68,00,00,05,74,00,00
1609 y5  DATA 05,C6,00,00,05,D2,00,00,05,E0
1610 mR  DATA 00,00,05,EE,00,00,05,FC,00,00
1611 5j  DATA 06,02,00,00,06,14,00,00,06,18
1612 7i  DATA 00,00,06,1E,00,00,06,3C,00,00
1613 r5  DATA 06,7A,00,00,06,84,00,00,06,8E
1614 RR  DATA 00,00,06,9C,00,00,06,A2,00,00
1615 Ov  DATA 06,B0,00,00,06,BE,00,00,06,D8
1616 A3  DATA 00,00,06,FE,00,00,07,22,00,00
1617 hR  DATA 07,36,00,00,07,3A,00,00,07,46
1618 30  DATA 00,00,07,50,00,00,07,5A,00,00
1619 kx  DATA 07,64,00,00,07,6E,00,00,07,A4
1620 yY  DATA 00,00,07,AE,00,00,07,B8,00,00
1621 ej  DATA 07,C2,00,00,07,CC,00,00,07,FE
1622 tw  DATA 00,00,08,08,00,00,08,12,00,00
1623 XT  DATA 08,1C,00,00,08,26,00,00,08,4C
1624 SC  DATA 00,00,08,56,00,00,08,60,00,00
1625 OF  DATA 08,6A,00,00,08,74,00,00,08,9A
1626 Db  DATA 00,00,08,A4,00,00,08,AA,00,00
1627 RA  DATA 08,B0,00,00,08,C8,00,00,08,D2
1628 Vp  DATA 00,00,08,DC,00,00,08,E6,00,00
```

```
1629 X2  DATA 08,F0,00,00,09,24,00,00,09,2E
1630 cT  DATA 00,00,09,38,00,00,09,42,00,00
1631 7N  DATA 09,74,00,00,09,7E,00,00,09,88
1632 yd  DATA 00,00,09,92,00,00,09,B6,00,00
1633 NL  DATA 09,C0,00,00,09,CA,00,00,09,D4
1634 Sg  DATA 00,00,09,FC,00,00,0A,02,00,00
1635 SM  DATA 0A,08,00,00,0A,20,00,00,0A,2A
1636 fE  DATA 00,00,0A,30,00,00,0A,3A,00,00
1637 5z  DATA 0A,48,00,00,0A,56,00,00,0A,60
1638 Ky  DATA 00,00,0A,6E,00,00,0A,88,00,00
1639 Ty  DATA 0A,8C,00,00,0A,92,00,00,0A,A4
1640 Uh  DATA 00,00,0A,AA,00,00,0A,BC,00,00
1641 p9  DATA 0A,C2,00,00,0A,D4,00,00,0A,EE
1642 sS  DATA 00,00,0B,08,00,00,0B,22,00,00
1643 1o  DATA 0B,3C,00,00,0B,AA,00,00,0B,B0
1644 1x  DATA 00,00,0B,BE,00,00,0B,C4,00,00
1645 PR  DATA 0B,CE,00,00,0B,E0,00,00,0B,EA
1646 OR  DATA 00,00,0C,00,00,00,0C,06,00,00
1647 k8  DATA 0C,0A,00,00,0C,10,00,00,0C,1A
1648 Iu  DATA 00,00,0C,24,00,00,0C,2C,00,00
1649 Mj  DATA 0C,3A,00,00,0C,44,00,00,0C,52
1650 e8  DATA 00,00,0C,6C,00,00,0C,86,00,00
1651 5U  DATA 0C,A0,00,00,0C,BA,00,00,0C,D4
1652 2Q  DATA 00,00,0C,EE,00,00,0D,08,00,00
1653 2I  DATA 0D,1A,00,00,0D,20,00,00,0D,32
1654 kJ  DATA 00,00,0D,38,00,00,0D,4A,00,00
1655 U5  DATA 0D,64,00,00,0D,6A,00,00,0D,82
1656 IO  DATA 00,00,0D,AC,00,00,0D,BC,00,00
1657 Yj  DATA 0D,CA,00,00,0D,D0,00,00,0D,D6
1658 Fr  DATA 00,00,0D,E4,00,00,0D,EA,00,00
1659 7E  DATA 0D,F0,00,00,0D,F6,00,00,0E,04
1660 Ps  DATA 00,00,0E,10,00,00,0E,1A,00,00
1661 e2  DATA 0E,48,00,00,0E,62,00,00,0E,6E
1662 mt  DATA 00,00,0E,78,00,00,0E,84,00,00
1663 ib  DATA 0E,94,00,00,0E,A4,00,00,0E,B6
1664 Ds  DATA 00,00,0E,C4,00,00,0E,CA,00,00
1665 Tt  DATA 0E,D0,00,00,0E,D6,00,00,0E,E4
1666 uQ  DATA 00,00,0E,EA,00,00,0E,F0,00,00
1667 Jv  DATA 0E,F6,00,00,0F,04,00,00,0F,0A
1668 pJ  DATA 00,00,0F,14,00,00,0F,1A,00,00
1669 On  DATA 0F,28,00,00,0F,2E,00,00,0F,48
1670 ri  DATA 00,00,0F,6C,00,00,10,0C,00,00
1671 Pv  DATA 10,3E,00,00,10,4E,00,00,10,84
1672 7N  DATA 00,00,10,8A,00,00,10,94,00,00
1673 QE  DATA 10,A6,00,00,10,B0,00,00,10,C6
1674 YF  DATA 00,00,10,CC,00,00,10,D0,00,00
1675 ov  DATA 10,D6,00,00,10,E0,00,00,10,EA
1676 SI  DATA 00,00,10,F2,00,00,11,00,00,00
1677 pE  DATA 11,0A,00,00,11,18,00,00,11,32
1678 2W  DATA 00,00,11,4C,00,00,11,66,00,00
1679 gr  DATA 11,80,00,00,11,9A,00,00,11,B4
1680 RG  DATA 00,00,11,CE,00,00,11,E8,00,00
1681 Eq  DATA 11,FA,00,00,12,00,00,00,12,12
1682 8h  DATA 00,00,12,18,00,00,12,2A,00,00
1683 zq  DATA 12,30,00,00,12,42,00,00,12,5C
1684 hv  DATA 00,00,12,62,00,00,12,7A,00,00
1685 ce  DATA 12,A4,00,00,12,B4,00,00,12,C2
1686 11  DATA 00,00,12,C8,00,00,12,CE,00,00
1687 xD  DATA 12,DC,00,00,12,E2,00,00,12,E8
1688 Ub  DATA 00,00,12,EE,00,00,13,00,00,00
1689 zH  DATA 13,0E,00,00,13,18,00,00,13,26
1690 xM  DATA 00,00,13,2C,00,00,13,32,00,00
1691 Bi  DATA 13,3E,00,00,13,44,00,00,13,4A
1692 ju  DATA 00,00,13,50,00,00,13,5A,00,00
1693 Eq  DATA 13,66,00,00,13,70,00,00,13,9A
1694 Sn  DATA 00,00,13,A0,00,00,13,B0,00,00
1695 Wp  DATA 13,CA,00,00,13,D0,00,00,13,E4
1696 iP  DATA 00,00,13,EA,00,00,13,FA,00,00
1697 3g  DATA 14,00,00,00,14,06,00,00,14,16
1698 g1  DATA 00,00,14,24,00,00,14,2A,00,00
1699 Ho  DATA 14,38,00,00,14,3E,00,00,14,48
1700 jI  DATA 00,00,14,4E,00,00,14,5C,00,00
1701 7p  DATA 14,62,00,00,14,70,00,00,14,76
1702 rz  DATA 00,00,14,86,00,00,14,9C,00,00
1703 NX  DATA 14,A6,00,00,14,B0,00,00,14,BA
1704 Z1  DATA 00,00,14,C4,00,00,15,3A,00,00
1705 KD  DATA 15,44,00,00,15,4E,00,00,15,58
1706 rd  DATA 00,00,15,86,00,00,15,90,00,00
1707 Ow  DATA 15,9A,00,00,15,A4,00,00,16,04
1708 26  DATA 00,00,16,16,00,00,16,20,00,00
1709 pQ  DATA 16,36,00,00,16,3C,00,00,16,40
1710 VQ  DATA 00,00,16,46,00,00,16,50,00,00
1711 YI  DATA 16,5A,00,00,16,62,00,00,16,70
1712 qv  DATA 00,00,16,7A,00,00,16,88,00,00
1713 dJ  DATA 16,A2,00,00,16,BC,00,00,16,D6
1714 4r  DATA 00,00,16,F0,00,00,17,0A,00,00
1715 cY  DATA 17,24,00,00,17,3E,00,00,17,58
1716 PM  DATA 00,00,17,6A,00,00,17,70,00,00
1717 Sh  DATA 17,82,00,00,17,88,00,00,17,9A
1718 cn  DATA 00,00,17,A0,00,00,17,B2,00,00
1719 tV  DATA 17,CC,00,00,17,D2,00,00,17,EA
1720 UQ  DATA 00,00,18,14,00,00,18,20,00,00
1721 qX  DATA 18,26,00,00,18,34,00,00,18,3A
1722 Ay  DATA 00,00,18,40,00,00,18,46,00,00
1723 OB  DATA 18,54,00,00,18,62,00,00,18,68
1724 fd  DATA 00,00,18,76,00,00,18,7C,00,00
1725 w0  DATA 18,86,00,00,18,8C,00,00,18,98
1726 JY  DATA 00,00,18,A2,00,00,18,A8,00,00
1727 uZ  DATA 18,B4,00,00,18,C4,00,00,18,CE
1728 Hu  DATA 00,00,18,DE,00,00,18,EE,00,00
1729 qh  DATA 18,F6,00,00,19,06,00,00,19,18
1730 k0  DATA 00,00,19,1E,00,00,19,26,00,00
1731 CF  DATA 19,36,00,00,19,3C,00,00,19,4C
1732 TQ  DATA 00,00,19,54,00,00,19,5A,00,00
1733 Oi  DATA 19,6A,00,00,19,84,00,00,19,8E
1734 V9  DATA 00,00,19,94,00,00,19,A2,00,00
1735 8J  DATA 19,A8,00,00,19,B8,00,00,19,C2
1736 ak  DATA 00,00,19,D0,00,00,19,DA,00,00
1737 Hc  DATA 19,E8,00,00,19,FE,00,00,1A,02
1738 JC  DATA 00,00,1A,08,00,00,1A,1A,00,00
1739 BW  DATA 1A,20,00,00,1A,32,00,00,1A,38
1740 7g  DATA 00,00,1A,4A,00,00,1A,50,00,00
1741 6j  DATA 1A,62,00,00,1A,68,00,00,1A,7A
1742 JR  DATA 00,00,1A,80,00,00,1A,92,00,00
1743 HL  DATA 1A,AC,00,00,1A,BC,00,00,1A,E2
1744 kR  DATA 00,00,1A,E8,00,00,1A,F6,00,00
1745 Ib  DATA 1A,FC,00,00,1B,00,00,00,1B,06
1746 R5  DATA 00,00,1B,10,00,00,1B,1A,00,00
1747 If  DATA 1B,22,00,00,1B,30,00,00,1B,3A
1748 Dr  DATA 00,00,1B,48,00,00,1B,5A,00,00
1749 8f  DATA 1B,64,00,00,1B,7A,00,00,1B,94
1750 17  DATA 00,00,1B,AE,00,00,1B,C8,00,00
1751 au  DATA 1B,E2,00,00,1B,FC,00,00,1C,0E
1752 Af  DATA 00,00,1C,14,00,00,1C,26,00,00
1753 cA  DATA 1C,2C,00,00,1C,3E,00,00,1C,58
1754 4e  DATA 00,00,1C,68,00,00,1C,8E,00,00
1755 U7  DATA 1C,94,00,00,1C,A2,00,00,1C,B0
1756 YS  DATA 00,00,1C,B6,00,00,1C,BC,00,00
1757 ln  DATA 1C,C6,00,00,1C,CA,00,00,1C,D4
1758 Bh  DATA 00,00,1C,E2,00,00,1C,E8,00,00
1759 Pg  DATA 1C,EE,00,00,1C,F8,00,00,1C,FC
1760 Ko  DATA 00,00,1D,02,00,00,1D,08,00,00
1761 HY  DATA 1D,14,00,00,1D,1A,00,00,1D,20
1762 Ir  DATA 00,00,1D,24,00,00,1D,2A,00,00
1763 zQ  DATA 1D,34,00,00,1D,3A,00,00,1D,44
1764 V5  DATA 00,00,1D,4E,00,00,1D,56,00,00
1765 A0  DATA 1D,60,00,00,1D,6A,00,00,1D,74
1766 yN  DATA 00,00,1D,7A,00,00,1D,88,00,00
1767 S5  DATA 1D,96,00,00,1D,B0,00,00,1D,B6
1768 Vc  DATA 00,00,1D,BC,00,00,1D,CE,00,00
1769 o9  DATA 1D,D4,00,00,1D,E6,00,00,1D,EC
1770 4J  DATA 00,00,1D,FE,00,00,1E,04,00,00
1771 Ti  DATA 1E,16,00,00,1E,30,00,00,1E,3A
1772 5J  DATA 00,00,1E,44,00,00,1E,72,00,00
1773 8E  DATA 1E,8C,00,00,1E,A6,00,00,1E,AC
1774 zV  DATA 00,00,1E,B6,00,00,1E,C0,00,00
1775 Di  DATA 1E,CA,00,00,1E,D4,00,00,1E,DE
1776 Kd  DATA 00,00,1E,E2,00,00,1E,F0,00,00
1777 T4  DATA 1E,F6,00,00,1F,00,00,00,1F,1E
1778 sL  DATA 00,00,1F,24,00,00,1F,2A,00,00
1779 kM  DATA 1F,56,00,00,1F,66,00,00,1F,E6
1780 sN  DATA 00,00,1F,F2,00,00,1F,FC,00,00
1781 AH  DATA 20,06,00,00,20,10,00,00,20,1A
1782 g0  DATA 00,00,20,20,00,00,20,26,00,00
1783 ZJ  DATA 20,30,00,00,20,34,00,00,20,3E
1784 Ht  DATA 00,00,20,48,00,00,20,5E,00,00
1785 k5  DATA 20,68,00,00,20,72,00,00,20,7C
1786 m9  DATA 00,00,20,86,00,00,20,8E,00,00
1787 rh  DATA 20,94,00,00,20,9A,00,00,20,A4
1788 OU  DATA 00,00,20,AA,00,00,20,BC,00,00
1789 fh  DATA 20,C6,00,00,20,D0,00,00,20,DA
1790 XD  DATA 00,00,20,E4,00,00,20,EC,00,00
1791 hg  DATA 20,F2,00,00,20,F8,00,00,21,04
1792 sS  DATA 00,00,21,10,00,00,21,2E,00,00
1793 1U  DATA 21,34,00,00,21,40,00,00,21,4C
1794 p6  DATA 00,00,21,6A,00,00,21,70,00,00
1795 8t  DATA 21,7C,00,00,21,88,00,00,21,A6
1796 2w  DATA 00,00,21,AC,00,00,21,B8,00,00
1797 EM  DATA 21,C4,00,00,21,E2,00,00,21,E8
1798 md  DATA 00,00,21,F4,00,00,22,00,00,00
1799 B1  DATA 22,1A,00,00,22,20,00,00,22,2C
1800 Xo  DATA 00,00,22,38,00,00,22,52,00,00
1801 GT  DATA 22,58,00,00,22,62,00,00,22,68
1802 xB  DATA 00,00,22,72,00,00,22,7C,00,00
1803 K7  DATA 22,84,00,00,22,92,00,00,22,9C
1804 vS  DATA 00,00,22,B2,00,00,22,C8,00,00
1805 yv  DATA 22,D6,00,00,22,FA,00,00,23,14
1806 Gm  DATA 00,00,23,2C,00,00,23,46,00,00
1807 4p  DATA 23,4C,00,00,23,5E,00,00,23,64
1808 DB  DATA 00,00,23,76,00,00,23,90,00,00
1809 r1  DATA 23,96,00,00,23,AE,00,00,23,D8
1810 Di  DATA 00,00,23,E4,00,00,23,EA,00,00
1811 ZB  DATA 23,F8,00,00,24,06,00,00,24,0C
1812 Zz  DATA 00,00,24,12,00,00,24,1C,00,00
1813 SL  DATA 24,20,00,00,24,26,00,00,24,2C
1814 Oj  DATA 00,00,24,3C,00,00,24,42,00,00
1815 WA  DATA 24,4E,00,00,24,5A,00,00,24,66
1816 nk  DATA 00,00,24,72,00,00,24,78,00,00
1817 8e  DATA 24,7E,00,00,24,84,00,00,24,92
1818 Zc  DATA 00,00,24,98,00,00,24,A6,00,00
1819 q8  DATA 24,B4,00,00,24,BA,00,00,24,C0
1820 Ga  DATA 00,00,24,C6,00,00,24,D0,00,00
1821 rC  DATA 24,DA,00,00,24,E2,00,00,24,F0
1822 73  DATA 00,00,24,FA,00,00,25,10,00,00
1823 b0  DATA 25,26,00,00,25,34,00,00,25,58
1824 im  DATA 00,00,25,72,00,00,25,8A,00,00
1825 S3  DATA 25,A4,00,00,25,AA,00,00,25,BC
1826 ev  DATA 00,00,25,C2,00,00,25,D4,00,00
1827 r1  DATA 25,EE,00,00,25,F4,00,00,26,0C
1828 WY  DATA 00,00,26,36,00,00,26,42,00,00
1829 7U  DATA 26,48,00,00,26,56,00,00,26,64
1830 Ws  DATA 00,00,26,6A,00,00,26,6E,00,00
1831 PG  DATA 26,74,00,00,26,7A,00,00,26,80
1832 XI  DATA 00,00,26,92,00,00,26,A0,00,00
1833 OY  DATA 26,A6,00,00,26,C6,00,00,26,D4
1834 nY  DATA 00,00,26,DE,00,00,26,EE,00,00
1835 1g  DATA 26,F4,00,00,27,00,00,00,27,0C
1836 ir  DATA 00,00,27,18,00,00,27,24,00,00
1837 1a  DATA 27,2C,00,00,27,32,00,00,27,36
1838 GS  DATA 00,00,27,3C,00,00,27,42,00,00
1839 WO  DATA 27,4C,00,00,27,50,00,00,27,5E
1840 tF  DATA 00,00,27,6E,00,00,27,7C,00,00
1841 Vy  DATA 27,9C,00,00,27,A8,00,00,27,AE
1842 Kd  DATA 00,00,27,B4,00,00,27,D6,00,00
1843 oC  DATA 27,DC,00,00,27,EC,00,00,28,06
1844 hx  DATA 00,00,28,2A,00,00,28,38,00,00
1845 4K  DATA 28,56,00,00,28,8C,00,00,28,AA
1846 8Z  DATA 00,00,28,CC,00,00,28,D6,00,00
1847 PM  DATA 28,E0,00,00,28,EA,00,00,28,F4
1848 pf  DATA 00,00,29,50,00,00,29,A0,00,00
1849 8y  DATA 29,B2,00,00,29,B8,00,00,29,BE
1850 mh  DATA 00,00,29,E0,00,00,29,F2,00,00
1851 wN  DATA 29,F8,00,00,29,FE,00,00,2A,18
1852 c3  DATA 00,00,2A,30,00,00,2A,44,00,00
1853 13  DATA 2A,4C,00,00,2A,66,00,00,2A,78
1854 iE  DATA 00,00,2A,7C,00,00,2A,82,00,00
1855 Rw  DATA 2A,98,00,00,2A,9E,00,00,2A,A6
1856 KO  DATA 00,00,2A,AC,00,00,2A,B4,00,00
1857 1k  DATA 2A,BA,00,00,2A,C2,00,00,2A,C8
1858 OA  DATA 00,00,2A,F4,00,00,2A,FA,00,00
1859 Dc  DATA 2B,22,00,00,2B,48,00,00,2B,96
1860 b5  DATA 00,00,2B,9C,00,00,2B,A2,00,00
1861 VI  DATA 2B,A8,00,00,2B,AE,00,00,2B,B4
1862 Gu  DATA 00,00,2B,C4,00,00,2B,D2,00,00
1863 9G  DATA 2B,DA,00,00,2B,E6,00,00,2B,F2
1864 Eo  DATA 00,00,2B,FE,00,00,2C,0A,00,00
1865 40  DATA 2C,12,00,00,2C,18,00,00,2C,38
1866 mI  DATA 00,00,2C,64,00,00,2C,6E,00,00
1867 K3  DATA 2C,82,00,00,2C,8C,00,00,2C,92
1868 ar  DATA 00,00,2C,98,00,00,2C,B2,00,00
1869 gx  DATA 2C,BA,00,00,2C,CC,00,00,2C,D2
1870 pn  DATA 00,00,2C,F2,00,00,2D,10,00,00
1871 pC  DATA 2D,14,00,00,2D,1C,00,00,2D,26
1872 ur  DATA 00,00,2D,2A,00,00,2D,D8,00,00
1873 Op  DATA 2D,F6,00,00,2E,08,00,00,2E,18
1874 Co  DATA 00,00,2E,2A,00,00,2E,46,00,00
1875 rZ  DATA 2E,4A,00,00,2E,88,00,00,2E,98
1876 PI  DATA 00,00,2E,AA,00,00,2E,C6,00,00
1877 mh  DATA 2E,CA,00,00,2F,3E,00,00,2F,50
1878 4J  DATA 00,00,2F,60,00,00,2F,6A,00,00
```

Listing 1. (Fortsetzung)

```
1879 VV  DATA 2F,6E,00,00,2F,9A,00,00,2F,AC
1880 nb  DATA 00,00,2F,BC,00,00,2F,C6,00,00
1881 ED  DATA 2F,CA,00,00,30,08,00,00,30,18
1882 Us  DATA 00,00,30,22,00,00,30,26,00,00
1883 KQ  DATA 30,58,00,00,30,68,00,00,30,72
1884 bu  DATA 00,00,30,76,00,00,30,A2,00,00
1885 Y1  DATA 30,B4,00,00,30,C4,00,00,30,CE
1886 x0  DATA 00,00,30,D2,00,00,31,10,00,00
1887 3y  DATA 31,20,00,00,31,2A,00,00,31,2E
1888 YD  DATA 00,00,31,5A,00,00,31,6C,00,00
1889 3m  DATA 31,7C,00,00,31,86,00,00,31,8A
1890 P5  DATA 00,00,31,B6,00,00,31,C8,00,00
1891 F4  DATA 31,D8,00,00,31,E2,00,00,31,E6
1892 xG  DATA 00,00,32,24,00,00,32,34,00,00
1893 xg  DATA 32,3E,00,00,32,42,00,00,33,DE
1894 vF  DATA 00,00,33,E2,00,00,33,E6,00,00
1895 Fr  DATA 33,EA,00,00,33,EE,00,00,33,F2
1896 wS  DATA 00,00,33,F6,00,00,33,FA,00,00
1897 Ad  DATA 33,FE,00,00,34,02,00,00,34,06
1898 YK  DATA 00,00,34,0A,00,00,34,0E,00,00
1899 tX  DATA 34,12,00,00,34,16,00,00,34,1A
1900 gB  DATA 00,00,34,1E,00,00,34,22,00,00
1901 L1  DATA 34,26,00,00,34,2A,00,00,34,2E
1902 eo  DATA 00,00,34,32,00,00,34,46,00,00
1903 rn  DATA 34,5A,00,00,34,6E,00,00,34,82
1904 A8  DATA 00,00,34,AA,00,00,34,BE,00,00
1905 Sn  DATA 34,D2,00,00,34,E6,00,00,35,0E
1906 jt  DATA 00,00,35,22,00,00,35,36,00,00
1907 uq  DATA 35,4A,00,00,35,5E,00,00,35,72
1908 7Q  DATA 00,00,35,86,00,00,35,AE,00,00
1909 zd  DATA 35,D6,00,00,35,EA,00,00,35,FE
1910 oy  DATA 00,00,36,12,00,00,36,26,00,00
1911 Eu  DATA 00,1D,00,00,00,01,00,00,0E,2C
1912 hE  DATA 00,00,0E,3E,00,00,0E,54,00,00
1913 Gm  DATA 0E,68,00,00,0E,AC,00,00,0F,40
1914 ag  DATA 00,00,0F,54,00,00,0F,64,00,00
1915 Ea  DATA 0F,78,00,00,0F,82,00,00,0F,A0
1916 Qu  DATA 00,00,0F,B8,00,00,0F,D0,00,00
1917 D7  DATA 13,7E,00,00,13,86,00,00,13,AA
1918 wH  DATA 00,00,13,B6,00,00,15,B6,00,00
1919 bQ  DATA 15,EE,00,00,1E,B0,00,00,1E,BA
1920 jU  DATA 00,00,1E,C4,00,00,1E,CE,00,00
1921 p4  DATA 1E,D8,00,00,1F,EE,00,00,1F,F8
1922 YD  DATA 00,00,20,02,00,00,20,0C,00,00
1923 Qm  DATA 20,16,00,00,00,00,00,00,03,F2
1924 w7  DATA 00,00,03,EB,00,00,01,00,00,00
1925 qX  DATA 03,F2
(C) 1987 M&T
```

Listing 1. (Schluß)

Fortsetzung von Seite 14

Heiße Schlachten im Amiga

Programmname:	Hauptprogramm
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2
Bemerkung:	Bitte zuerst das Ladeprogramm (Listing 2) starten, damit genügend Speicher bereitsteht

```
Programm : hauptprogramm
------------------------

  1 hTO SCREEN 1,640,255,3,2
  2 IX  WINDOW 2,"Amiga gegen den Rest der Welt",(0,0)-(615,230),16
        ,1
  3 Va  REM Weltkarte
  4 M1  f=1:z=0:start=0:FOR p=1 TO 6:ka(p)=0:NEXT
  5 qp  za=3:abbruch=0:ab=0:dugmax=7
  6 Nw  GOTO spielabfr
  7 hk  namen:
  8 yG2   PALETTE 3,.93,.2,0:REM rot
  9 iP    PALETTE 4,1,1,.1:REM gelb
 10 yW    PALETTE 5,1,.6,.3:REM orange
 11 Xb    PALETTE 6,.33,.87,0:REM GRÜN
 12 vz    PALETTE 7,1,.7,.9:REM rosa
 13 kw    REM weiß=1
 14 IS    LOCATE 23,19:PRINT "Abbruch"
 15 UN    LINE(210,175)-(280,185),1,B:LINE(210,175)-(280,185),1
 16 yd    LINE(210,185)-(280,175),1
 17 Ck    PAINT(212,180),3,1:PAINT(258,180),3,1
 18 5g    PAINT(235,176),5,1:PAINT(235,184),5,1
 19 9L    LOCATE 21,20:PRINT "Spieler"
 20 p8    LINE(210,157)-(280,170),1,B
 21 9S    COLOR 2,0
 22 LP    LOCATE 3,2:PRINT "Alaska":LOCATE 2,10:PRINT"NW-"
 23 Jc    LOCATE 3,10:PRINT"Territorium"
 24 ui    LOCATE 2,26:PRINT"Grönland"
 25 N5    LOCATE 6,6:PRINT"Alberta"
 26 TS    LOCATE 6,20:PRINT"Ontario"
 27 Jg    LOCATE 5,26:PRINT"Quebeck"
 28 Xd    LOCATE 8,2:PRINT"Weststaaten"
 29 Nu    LOCATE 9,20:PRINT"Oststaaten"
 30 41    LOCATE 11,2:PRINT"Mittel-":LOCATE 12,4:PRINT"amerika"
 31 Ea    LOCATE 14,2:PRINT"Venezuela"
 32 46    LOCATE 17,5:PRINT"Peru"
 33 1u    LOCATE 17,14:PRINT"Brasilien"
 34 NM    LOCATE 22,4:PRINT"Argentinien"
 35 JU    LOCATE 6,31:PRINT"Island"
 36 1j    LOCATE 5,37:PRINT"Skandinavien"
 37 qI    LOCATE 10,31:PRINT"GB"
 38 8x    LOCATE 7,47:PRINT"Ukraine"
 39 AL    LOCATE 6,55:PRINT"Ural"
 40 7v    LOCATE 3,60:PRINT"Sibirien"
 41 pU    LOCATE 2,68:PRINT"Jakutien"
 42 Ta    LOCATE 6,67:PRINT"Irkutsk"
 43 HI    LOCATE 10,68:PRINT"Mongolei"
 44 Ep    LOCATE 7,73:PRINT"Kamt-":LOCATE 8,71:PRINT"schatka"
 45 0j    LOCATE 13,60:PRINT"China"
 46 U6    LOCATE 15,58:PRINT"Indien"
 47 eW    LOCATE 13,46:PRINT"Mittlerer":LOCATE 14,47:PRINT"Osten"
 48 J2    LOCATE 11,54:PRINT"Afghanistan"
 49 NB    LOCATE 15,66:PRINT"Siam"
 50 EU    LOCATE 17,72:PRINT"Neu":LOCATE 18,72:PRINT"Guinea"
 51 GV    LOCATE 19,59:PRINT"Indonesien"
 52 sY    LOCATE 22,59:PRINT"West-":LOCATE 23,60:PRINT"Australien"
 53 1s    LOCATE 22,73:PRINT"Ost-":LOCATE 25,68:PRINT"Australien"
 54 7P    LOCATE 11,72:PRINT"Japan"
 55 1Z    LOCATE 15,28:PRINT"Nord-West-":LOCATE 17,31:PRINT"Afrika"
 56 s5    LOCATE 15,40:PRINT"Ägypten"
 57 Oz    LOCATE 17,45:PRINT"Ostafrika"
 58 eG    LOCATE 19,42:PRINT"Kongo"
 59 6m    LOCATE 23,37:PRINT"SÜdafrika"
 60 OY    LOCATE 24,52:PRINT"Madagaskar"
 61 Lx    RETURN
 62 tS    REM alaska
 63 cM0 a1:
 64 pe4     LOCATE 4,5:PRINT land(L)
 65 6o      LINE(30,4)-(32,14):LINE-(48,9):LINE-(64,13)
 66 4h      LINE-(72,16):LINE-(72,40):LINE-(48,35)
 67 ux      LINE-(40,36)
 68 H1      LINE-(24,33):LINE-(16,25):LINE-(24,22)
 69 XA      LINE-(16,15):LINE-(48,11)
 70 91      LINE-(30,4)
 71 8N      PAINT (33,15),f,2
 72 W8      RETURN
 73 bt2   REM Nordwest-Territorium
 74 qb0 a2:
 75 9I4     LOCATE 4,15:PRINT land(L)
 76 3s      LINE(72,16)-(128,14):LINE-(176,28)
 77 BW      LINE-(176,35):LINE-(96,35)
 78 GE      LINE-(72,40)
 79 Vn      PAINT (75,30),f,2
 80 eG      RETURN
 81 SQ2   REM Grönland
 82 1n0 a3:
 83 JG4     LOCATE 3,32:PRINT land(L)
 84 PN      LINE(176,28)-(210,15):LINE-(230,20)
 85 vT      LINE-(200,35):LINE-(176,35)
 86 dv      LINE(248,14)-(250,5):LINE-(270,6)
 87 Nq      LINE-(288,20):LINE-(260,40)
 88 dT      LINE-(248,25):LINE-(248,14)
 89 4U      PAINT (205,30),f,2
 90 jN      PAINT(249,15),f,2
 91 pR      RETURN
 92 vH3    REM Alberta
 93 F20 a4:
 94 Yi4     LOCATE 6,14:PRINT land(L)
 95 50      LINE(72,40)-(72,55):LINE-(160,55)
 96 KE      LINE-(160,35)
 97 yf      PAINT(159,36),f,2
 98 wY      RETURN
 99 S63    REM Ontario
100 PD0 a5:
101 1q5      LOCATE 7,22:PRINT land(L)
102 vm       LINE(200,35)-(210,60):LINE-(180,55)
103 VR       LINE-(160,55)
104 T3       PAINT(161,54),f,2
105 3f       RETURN
106 Z53    REM Quebeck
107 Z00 a6:
108 2C5      LOCATE 7,27:PRINT land(L)
```

```
04       LINE(202,50)-(225,40):LINE-(245,53)
AG       LINE-(225,58):LINE-(210,60):LINE-(200,35)
10       PAINT(224,57),f,2
Am       RETURN
iY3    REM Weststaaten
jZ0 a7:
5I5      LOCATE 9,14:PRINT land(L)
QS       LINE(72,55)-(70,65):LINE-(73,72)
nx       LINE-(80,75):LINE-(130,80)
j9       LINE-(145,60):LINE-(170,55)
CM       PAINT(72,60),f,2
Iu       RETURN
sw3    REM Oststaaten
ul0 a8:
Ec5      LOCATE 10,19:PRINT land(L)
30       LINE(130,80)-(140,85):LINE-(170,80)
Kf       LINE-(172,90):LINE-(180,88)
Xt       LINE-(179,80):LINE-(215,70)
pl       LINE-(220,58)
zZ       PAINT(218,60),f,2
R3       RETURN
2d3    REM Mittelamerika
6y0 a9:
4L5      LOCATE 11,11:PRINT land(L)
MH       LINE(73,72)-(74,75):LINE-(85,90)
h1       LINE-(90,98):LINE-(112,105)
kB       LINE-(116,104):LINE-(105,95)
gn       LINE-(110,91):LINE-(120,87)
1y       LINE-(140,85)
qP       PAINT(130,84),f,2
bD       RETURN
pJ3    REM Venezuela
KK0 a10:
bx5      LOCATE 15,12:PRINT land(L)
sH       LINE(116,104)-(128,103):LINE-(180,123)
xm       LINE-(148,124):LINE-(118,119)
Nr       LINE-(94,124):LINE-(66,119)
Ww       LINE-(76,111):LINE-(100,105)
BG       LINE-(112,105)
4C       PAINT(90,122),f,2
1N       RETURN
qg3    REM Preu
YZ0 a11:
zN5      LOCATE 19,10:PRINT land(L)
vE       LINE(66,119)-(56,132):LINE-(64,146)
AU       LINE-(76,154):LINE-(78,159)
Ds       LINE-(114,164):LINE-(122,144)
g5       LINE-(88,133):LINE-(94,124)
Wy       PAINT(112,162),f,2
uW       RETURN
Hk3    REM Brasilien
1n0 a12:
W25      LOCATE 19,18:PRINT land(L)
Nv       LINE(122,144)-(136,152):LINE-(134,165)
80       LINE-(140,169):LINE-(164,156)
6v       LINE-(186,154):LINE-(216,142)
jt       LINE-(180,123)
tP       PAINT(140,165),f,2
3f       RETURN
7E3    REM Argentinien
y10 a13:
uE5      LOCATE 23,11:PRINT land(L)
yW       LINE(78,159)-(80,178):LINE-(74,194)
z0       LINE-(76,208):LINE-(84,215)
h1       LINE-(92,219):LINE-(90,213)
rJ       LINE-(92,208):LINE-(98,201)
DT       LINE-(140,169)
6H       PAINT(88,213),f,2
Dp       RETURN
ZX3    REM Island
CG0 a14:
AC5      LOCATE 7,33:PRINT land(L)
eB       LINE(255,44)-(273,48):LINE-(283,42)
Aj       LINE-(303,51):LINE-(279,58)
Hp       LINE-(261,59):LINE-(248,50)
76       LINE-(255,44)
tW       PAINT(260,57),f,2
My       RETURN
h73    REM Skandinavien
PU0 a15:
Zi5      LOCATE 8,39:PRINT land(L)
ar       LINE(292,72)-(290,64):LINE-(306,57)
jB       LINE-(314,48):LINE-(312,41)
zc       LINE-(326,38):LINE-(340,35)
Mr       LINE-(350,47):LINE-(354,67)
Eq       LINE-(346,61):LINE-(342,48)
1J       LINE-(336,54):LINE-(338,66)
tA       LINE-(330,74):LINE-(320,76)
sM       LINE-(308,69):LINE-(292,72)
oN       PAINT(320,74),f,2
ZB       RETURN
Um3    REM Mitteleuropa
gm0 a16:
b15      LOCATE 11,38:PRINT land(L)
oP       LINE(288,87)-(298,83):LINE-(306,75)
Yp       LINE-(312,76):LINE-(318,81)
tH       LINE-(346,75):LINE-(360,85)
uK       LINE-(312,91):LINE-(288,87)
vT       PAINT(312,90),f,2
iK       RETURN
KH3    REM Westeuropa
t00 a17:
1A5      LOCATE 13,35:PRINT land(L)
Rb       LINE(288,87)-(284,92):LINE-(276,95)
Mo       LINE-(280,97):LINE-(264,99)
aX       LINE-(268,107):LINE-(286,108)
ZA       LINE-(288,101):LINE-(312,98)
C7       LINE-(312,91)
tA       PAINT(284,107),f,2:PAINT(288,89),f,2
sU       RETURN
rZ3    REM SÜdeuropa
7F0 a18:
j55      LOCATE 12,42:PRINT land(L)
Zn       LINE(312,98)-(322,103):LINE-(314,106)
Lz       LINE-(332,107):LINE-(334,103)
Pm       LINE-(328,96):LINE-(340,97)
xc       LINE-(352,102):LINE-(358,99)
Yy       LINE-(360,95):LINE-(368,95)
Kw       LINE-(360,85):LINE(360,85)-(312,91)
h9       PAINT(330,106),f,2
3f       RETURN
Ow3    REM Ukraine
MV0 a19:
QW5      LOCATE 9,49:PRINT land(L)
dR       LINE(368,95)-(370,98):LINE-(414,109)
Sz       LINE-(412,106):LINE-(412,99)
EZ       LINE-(416,97):LINE-(422,81)
2U       LINE-(434,80):LINE-(446,62)
Tv       LINE-(426,52):LINE-(420,42)
JG       LINE-(438,29):LINE-(428,16)
ud       LINE-(380,28):LINE-(380,38)
L6       LINE-(366,37):LINE-(356,33)
9W       LINE-(340,35):LINE(346,75)-(354,67)
bC       PAINT(396,103),f,2
Ht       RETURN
xP3    REM Groß Britanien
340 a20:
Mm5      LOCATE 12,28:PRINT land(L)
7x       LINE(256,75)-(262,75):LINE-(268,81)
De       LINE-(278,88):LINE-(272,92)
hz       LINE-(262,93):LINE-(258,90)
Qt       LINE-(260,85):LINE-(254,78)
QU       LINE-(256,75)
Oy       LINE(240,81)-(248,82):LINE-(254,87)
Yo       LINE-(252,91):LINE-(246,94)
oE       LINE-(234,89):LINE-(236,84)
mg       LINE-(240,81)
d7       PAINT(263,92),f,2:PAINT(246,93),f,2
V7       RETURN
Ib3    REM Ural
LN0 a21:
qp5      LOCATE 8,57:PRINT land(L)
Of       LINE(428,16)-(458,13):LINE-(468,4)
Vm       LINE-(474,3):LINE-(478,5)
HA       LINE-(480,22):LINE-(486,18)
Fq       LINE-(492,24):LINE-(496,78)
It       LINE-(486,79):LINE-(480,72)
2c       LINE-(456,66):LINE-(446,62)
C4       PAINT(488,78),f,2
gI       RETURN
Yt3    REM Afghanistan
ad0 a22:
1C5      LOCATE 12,56:PRINT land(L)
E9       LINE(486,79)-(470,95):LINE-(462,107)
```

Listing 1. (Fortsetzung)

```
sg       LINE-(448,113):LINE-(416,107)
nt       LINE-(416,97)
sP       PAINT(448,110),f,2
oQ       RETURN
513    REM Mittlerer Osten
mq0  a23:
F15      LOCATE 15,49:PRINT land(L)
MC       LINE-(448,113)-(448,118):LINE-(436,118)
jU       LINE-(416,113):LINE-(424,119)
su       LINE-(398,128):LINE-(394,124)
2r       LINE-(374,115):LINE-(374,105)
MO       LINE-(356,103):LINE-(362,100)
sy       LINE-(370,98)
Wx       PAINT(400,125),f,2
zb       RETURN
p43    REM Sibirien
160  a24:
6u5      LOCATE 5,63:PRINT land(L)
LV       LINE(492,24)-(486,18):LINE-(484,9)
me       LINE-(500,8):LINE-(510,3)
8B       LINE-(546,1):LINE-(530,29)
Jt       LINE-(534,47):LINE-(522,78):LINE-(506,80)
dq       LINE-(496,78)
Mv       PAINT(500,74),f,2
91       RETURN
EI3    REM Jakutien
FL0  a25:
KC5      LOCATE 3,69:PRINT land(L)
hG       LINE(546,1)-(560,6):LINE-(580,3):LINE-(584,23)
m6       LINE-(578,30):LINE-(564,38):LINE-(560,28)
pn       LINE-(530,29)
sT       PAINT(564,30),f,2
Ht       RETURN
Cn3    REM Irkutsk
RY0  a26:
gb5      LOCATE 7,68:PRINT land(L)
xf       LINE(522,78)-(534,76):LINE-(554,71):LINE-(550,63)
TJ       LINE-(564,54):LINE-(564,38)
oQ       PAINT(530,65),f,2
00       RETURN
Al3    REM Kamtschatka
ck0  a27:
bL5      LOCATE 5,73:PRINT land(L)
sI       LINE(580,3)-(594,6):LINE-(606,13):LINE-(602,18)
mS       LINE-(612,26):LINE-(610,36):LINE-(594,51)
tm       LINE-(598,66):LINE-(592,69):LINE-(584,62)
J5       LINE-(584,54):LINE-(574,65):LINE-(566,71)
Al       LINE-(570,79):LINE-(562,86):LINE-(554,71)
Ds       PAINT(552,64),f,2
YA       RETURN
Gp3    REM Mongolei
qz0  a28:
d55      LOCATE 11,67:PRINT land(L)
H1       LINE(562,86)-(568,92):LINE-(552,91):LINE-(526,87)
lw       LINE-(506,80):LINE(562,86)-(554,71)
wV       PAINT(512,80),f,2
fH       RETURN
9f3    REM China
1B0  a29:
oG5      LOCATE 13,65:PRINT land(L)
K7       LINE(462,107)-(482,107):LINE-(510,119):LINE-(528,114)
qH       LINE-(552,116):LINE-(548,122):LINE-(560,119)
PQ       LINE-(566,108):LINE-(560,98):LINE-(552,91)
6w       PAINT(494,79),f,2
nP       RETURN
MS3    REM Indien
ce0  a30:
wM5      LOCATE 16,60:PRINT land(L)
I3       LINE(448,118)-(456,120):LINE-(470,132):LINE-(480,140)
p1       LINE-(490,131):LINE-(510,125):LINE-(510,119)
Cp       PAINT(480,138),f,2
uW       RETURN
Ha3    REM Siam
nq0  a31:
In5      LOCATE 16,65:PRINT land(L)
V2       LINE(510,125)-(524,130):LINE-(530,133):LINE-(530,129)
3V       LINE-(546,141):LINE-(544,136):LINE-(550,137)
Sy       LINE-(560,134):LINE-(558,136):LINE-(548,122)
9J       PAINT(542,136),f,2
2e       RETURN
Wa3    REM Japan
z30  a32:
Ac5      LOCATE 13,74:PRINT land(L)
zx       LINE(582,81)-(588,88):LINE-(586,94):LINE-(592,103)
```

```
Sp       LINE-(582,108):LINE-(572,104):LINE-(574,98)
Bq       LINE-(576,92):LINE-(574,84):LINE-(582,81)
En       PAINT(580,106),f,2
Am       RETURN
7V3    REM Neu Guinea
BG0  a33:
mK5      LOCATE 19,74:PRINT land(L)
87       LINE(574,137)-(584,137):LINE-(602,139):LINE-(610,144)
M6       LINE-(614,159):LINE-(594,157):LINE-(580,148)
jg       LINE-(574,141):LINE-(574,137)
Gp       PAINT(610,157),f,2
Iu       RETURN
3r3    REM Indonesien
NT0  a34:
7j5      LOCATE 19,69:PRINT land(L)
nw       LINE(564,141)-(572,146):LINE-(576,152):LINE-(564,159)
1c       LINE-(538,157):LINE-(532,153):LINE-(534,149)
C0       LINE-(540,145):LINE-(564,141)
2u       LINE(514,150)-(524,152):LINE-(534,159):LINE-(528,161)
yI       LINE-(518,160):LINE-(514,154):LINE-(514,150)
Wr       PAINT(564,158),f,2:PAINT(530,159),f,2
S46       RETURN
Et3    REM Westaustralien
bi0  a35:
jE5      LOCATE 22,68:PRINT land(L)
mu       LINE(572,163)-(558,162):LINE-(542,167):LINE-(532,168)
XG       LINE-(516,179):LINE-(522,188):LINE-(536,193)
8z       LINE-(552,190):LINE-(568,185):LINE-(578,177)
14       LINE-(572,163)
6m       PAINT(538,191),f,2
bD       RETURN
OT3    REM Ostaustralien
ow0  a36:
nG5      LOCATE 23,74:PRINT land(L)
w5       LINE(572,163)-(582,159):LINE-(606,166):LINE-(614,174)
wJ       LINE-(610,184):LINE-(598,191):LINE-(568,185)
tW       PAINT(600,189),f,2
iK       RETURN
ad3    REM Nord-West-Afrika
z80  a37:
Ae5      LOCATE 18,35:PRINT land(L)
7J       LINE(308,112)-(304,110):LINE-(272,110):LINE-(258,117)
mx       LINE-(250,130):LINE-(260,144):LINE-(284,150)
OS       LINE-(300,149):LINE-(308,157):LINE-(318,156)
Vp       LINE-(326,147):LINE-(340,144):LINE-(328,140)
Bd       LINE-(332,135):LINE-(298,125):LINE-(296,121)
gq       LINE-(308,112)
qN       PAINT(262,143),f,2
tV       RETURN
4u3    REM Ägypten
E00  a38:
4U5      LOCATE 16,42:PRINT land(L)
5f       LINE(308,112)-(334,114):LINE-(352,113):LINE-(374,115)
XG       LINE-(380,124):LINE-(332,135)
lF       PAINT(332,133),f,2
0c       RETURN
DK3    REM Ostafrika
Pa0  a39:
Uz5      LOCATE 18,45:PRINT land(L)
2w       LINE(380,124)-(384,127):LINE-(396,130):LINE-(406,129)
VG       LINE-(394,145):LINE-(398,157):LINE-(386,170)
1R       LINE-(352,163):LINE-(356,155):LINE-(350,145)
p0       LINE-(340,144)
tc       PAINT(386,166),f,2
91       RETURN
Yr3    REM Kongo
140  a40:
rB5      LOCATE 20,41:PRINT land(L)
N6       LINE(352,163)-(332,163):LINE-(320,160):LINE-(308,159)
Zs       LINE-(308,157)
8r       LINE(300,157)-(318,156):LINE-(326,147)
5b       PAINT(350,162),f,2
Ht       RETURN
Ma3    REM SÜdafrika
DH0  a41:
6R5      LOCATE 22,40:PRINT land(L)
72       LINE(308,159)-(314,173):LINE-(312,182):LINE-(326,195)
Hf       LINE-(342,206):LINE-(354,202):LINE-(368,191)
i2       LINE-(386,170)
Bf       PAINT(342,204),f,2
P1       RETURN
Rj3    REM Madagaskar
PU0  a42:
Om5      LOCATE 23,51:PRINT land(L)
```

```
IO      LINE(410,171)-(414,176):LINE-(412,185):LINE-(400,193)
lK      LINE-(392,193):LINE-(390,187):LINE-(402,173)
1A      LINE-(410,171)
09      PAINT(398,192),f,2
X9      RETURN
zr3    REM *** Spieleranzahl ***
KE1  spielabfr:
Tp5     INPUT"Wieviele menschliche Spieler (0-6)";sp
d07       IF sp=6 THEN GOTO w1
II        IF sp>6 THEN PRINT"zu viel !!!":GOTO spielabfr
c05     PRINT"Wieviele Spieler soll der Computer Übernehmen (m
        ax.";6-sp;")";:INPUT" ",csp
6v7       IF csp+sp>6 THEN csp=6-sp
ci3    w1:
N16      spges=sp+csp:z=1
nX       FOR I=1 TO spges
529         IF I>sp THEN Spieler$(I)=CHR$(70+z):z=z+1
UZ          IF I<sp+1 THEN Spieler$(I)=CHR$(64+I)
TY6      NEXT
aH2   DIM land(43),land$(43)
nv    FOR I=1 TO 42:land(I)=1:land$(I)="*":NEXT
cl5     RANDOMIZE TIMER
QU2   FOR D=1 TO spges
eF3    landanz=INT(42/spges)
MU     FOR dd=1 TO landanz
qF     rndw:
wI4     x=INT(RND*44):IF x>42 THEN x=42
aU      IF land$(x)<>"*" THEN GOTO rndw
v2      land$(x)=Spieler$(D)
Fj3    NEXT:NEXT
zB5     z=1
vc3    FOR I=1 TO 42
Cv     IF land$(I)="*" THEN land$(I)=Spieler$(z):z=z+1
in     NEXT
3B2   w3:
38    IF sp=0 THEN GOTO w4
wk4     PRINT :PRINT"Bitte nacheinander die Namen eingeben"
80      PRINT "    Höchstens 8 Buchstaben bitte !"
bd2   FOR D=1 TO sp
Ps4     LOCATE 5+D,5:LINE INPUT N$(D)
pu      NEXT
3y2   LOCATE 14,2
tk    PRINT"Farbtabelle:"
D94     LOCATE 16,1
Da      PRINT "weiß=1  rot=3  gelb=4  orange=5  grÜn=6  rosa=7"
r05      LINE(0,118)-(390,128),,B:LINE(55,118)-(55,128)
16       LINE(110,118)-(110,128):LINE(175,118)-(175,128)
SX       LINE(255,118)-(255,128):LINE(320,118)-(320,128)
mG4     LOCATE 18,5:PRINT"Bitte eine Nummer wählen !"
mo2   FOR D=1 TO sp
la3    LOCATE 19+D,1
7j     PRINT N$(D);:INPUT" welche Farbe";farb(D)
Wk5      IF farb(D)=1 THEN LOCATE 16,1:PRINT "      "
Bj       IF farb(D)=3 THEN LOCATE 16,9:PRINT "     "
mD       IF farb(D)=4 THEN LOCATE 16,16:PRINT "      "
ZK       IF farb(D)=5 THEN LOCATE 16,24:PRINT"        "
sV       IF farb(D)=6 THEN LOCATE 16,34:PRINT "      "
pc       IF farb(D)=7 THEN LOCATE 16,42:PRINT "      "
7C3    NEXT
Sj     IF sp=6 OR csp=0 THEN GOTO w5
Jj     REM *Computer-Namen*
Xg2   w4:
hr5      FOR D=sp+1 TO spges
hS       READ cname$
QQ       N$(D)=cname$
EJ       NEXT
Gn       DATA "Amiga","Atari","Napoleon","Hirni"
OV       DATA "007","Fred"
tU3    REM *Computer-Farben*
6c5      IF sp=0 THEN GOTO far2
79       FOR D=1 TO sp
R2       farb2(D)=farb(D)
LQ       NEXT
ep       y=1
GD       RESTORE fdaten
tq3     falesen:
QN5      READ fa:z=0
Kl       IF fa=20 THEN GOTO w6
FH       FOR D=1 TO sp
RO       IF fa<>farb2(D) THEN z=z+1
Q3       IF z=sp THEN farb3(y)=fa:y=y+1
Xm       NEXT D:z=0:GOTO falesen
OF4      fdaten:
Mw7         DATA 1,3,4,5,6,7,20
7E4      far2:
R06        RESTORE fdaten
Iy         FOR p=1 TO spges:READ fa
7J         farb(p)=fa:NEXT
oL         GOTO nbl
5G0  w6:
v65       y=1
AK        FOR D=sp+1 TO spges
Wx        farb(D)=farb3(y):y=y+1
gl        NEXT
7H0  w5:
vL1   REM Nachbarländer einlesen
MK    nbl:
8H    DIM Nb(42,8),Lname$(42)
i03     RESTORE Nachbarland
Jo      FOR L=1 TO 42
qn5       FOR p=1 TO 8
eu        READ N
Fl        Nb(L,p)=N
q7        NEXT p,L
Uq2     RESTORE mausposition
Pu4      FOR L=1 TO 42
sS6        READ A,B,c,c,L$
bM         Lname$(L)=L$
v0         NEXT
pQ2     REM *** Reihenfolge ***
f85       FOR p=1 TO spges:rf$(p)="+":NEXT
7L2     FOR p=1 TO spges
cd      rfg:
6F      RANDOMIZE TIMER
ll      x=INT(RND(1)*spges+1)
qQ      IF x=0 THEN GOTO rfg
w0      IF x>spges THEN x=spges
6G4       IF rf$(x)<>"+" THEN GOTO rfg
15        rf$(x)=Spieler$(p)
6B        NEXT
RV2     WINDOW 4,,(250,130)-(400,135),0,1
Yq      WINDOW OUTPUT 2
ue      CLS:GOSUB namen:beg=0
hC      FOR L=1 TO 42
Cf4       IF land$(L)=Spieler$(1) THEN f=farb(1)
Jo        IF land$(L)=Spieler$(2) THEN f=farb(2)
Qx        IF land$(L)=Spieler$(3) THEN f=farb(3)
X6        IF land$(L)=Spieler$(4) THEN f=farb(4)
eF        IF land$(L)=Spieler$(5) THEN f=farb(5)
10        IF land$(L)=Spieler$(6) THEN f=farb(6)
Am        GOSUB v6
NW3     NEXT:GOTO v7
ku0  v6:
k23     WINDOW OUTPUT 2
34      ON L GOSUB a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a1
        4,a15,a16,a17,a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a2
        8,a29,a30,a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,a41,a4
        2
Lf5       GOSUB f4
Vr        IF beg=0 AND L=1 THEN CLS:PRINT " Nordamerika";
Ua        IF beg=0 AND L=9 THEN CLS:PRINT " SÜdamerika";:GOSUB p
          ause
QJ        IF beg=0 AND L=13 THEN CLS:PRINT " Europa";:GOSUB paus
          e
4C        IF beg=0 AND L=20 THEN CLS:PRINT " A s i e n";:GOSUB p
          ause
5q        IF beg=0 AND L=32 THEN CLS:PRINT " Australien";:GOSUB
          pause
rr        IF beg=0 AND L=36 THEN CLS:PRINT " A f r i k a";:beg=1
kM3     RETURN
942     END
zA0  v7:
sn1   REM   **** D A S   S P I E L   B E G I N N T ****
jQ4     WINDOW CLOSE 3
072     MENU 1,0,1,"Aktionen        "
7w      MENU 1,1,1,"Angriff"
9c      MENU 1,2,1,"Verschieben"
18      MENU 1,3,1,"Tauschen"
EV      MENU 1,4,1,"nächster Spieler"
GR      MENU 1,5,1,"ende"
17      DIM leer$(80),vs(42)
6A1   start:
```

Listing 1. (Fortsetzung)

```
WINDOW 4,,(0,200)-(620,235),0,1
WINDOW OUTPUT 4
r=1:dug=1
beginn:
  Karte=0:vt=1:leer$="
           "
  GOSUB f4
  GOSUB welcherSpieler
  GOSUB wievielArmeen
IF rf$(r)>"F" THEN GOTO computer
spiel:
  ON MENU GOSUB Spieler:MENU ON
  GOTO spiel
Spieler:
  v=MENU(1):CLS
  ON v GOSUB angriff,verschieben,tauschen,nextSpieler,en
  de
  GOSUB f4
  RETURN
  angriff:
    GOSUB f4:PRINT "Von"
  angr1:
    GOSUB welchesLand
      IF abbruch=1 THEN GOSUB f4:CLS:abbruch=0:RETURN
    IF land$(L)<>rf$(r) THEN
      CLS:LOCATE 3,5
      PRINT"Dies ist nicht Dein Land ! Noch mal.":GOTO a
      ngriff
      ELSE
      END IF
    IF vt=1 THEN RETURN
      IF land(L)<2 THEN
        CLS:LOCATE 2,5
        PRINT"Hier gibt es keine Angriffsarmeen !"
        GOSUB pause:CLS:GOTO angriff
        ELSE
        END IF
    AusgLand$=la$:Anr=L
    LOCATE 1,1
    PRINT "Von ";AusgLand$;" nach "
  ziel:
    GOSUB welchesLand
      IF abbruch=1 THEN GOSUB f4:CLS:abbruch=0:RETURN
    IF land$(L)=rf$(r) THEN
      LOCATE 3,5
      PRINT"Dies ist Dein eigenes Land ! Noch mal.";:GOT
      O ziel
      ELSE
      END IF
        FOR p=1 TO 8
          IF L=Nb(Anr,p) THEN GOTO z2
        NEXT
        LOCATE 3,5
        PRINT "Dies ist kein Nachbarland ! Noch mal.";:G
        OTO ziel
   z2:
    ZielLand$=la$:Znr=L:
    LOCATE 3,5:PRINT "
         "
    LOCATE 1,1
    PRINT "Von ";AusgLand$;" nach ";ZielLand$
    FOR p=1 TO spges
      IF Spieler$(p)=land$(Znr) THEN
        Znamen$=N$(p):fz=farb(p)
        ELSE
        END IF
      NEXT
    z4:
    FOR I=1 TO 42:vs(I)=0:NEXT
    z=0:GOSUB wuerfel:GOSUB auswertung
      IF ka(spi)=6 THEN
        CLS:PRINT "Du hast 6 Karten und mußt tauschen !"

        GOSUB pause:GOSUB wKarte
        ELSE
        END IF
      IF land(Anr)<2 OR land$(Znr)=rf$(r) THEN RETURN
    LOCATE 3,2
    PRINT "Soll der Angriff wiederholt werden ?";
    LOCATE 5,37:PRINT "Ja    Nein";
    LINE(280,30)-(310,40),,B:LINE(330,30)-(370,40),,B
    z3:
      m=MOUSE(0):IF MOUSE(0)=0 THEN z3
      IF MOUSE(3)>280 AND MOUSE(3)<315 THEN
```

```
        CLS:GOSUB pause
        LOCATE 1,1:PRINT "Von ";AusgLand$;" nach ";ZielL
        and$
        GOTO z4
        ELSE
        END IF
        CLS:RETURN
welchesLand:
  m=MOUSE(0):IF MOUSE(0)=0 THEN GOTO welchesLand
  hori=MOUSE(3):verti=MOUSE(4)
  LOCATE 3,1:PRINT leer$;
  RESTORE mausposition
  FOR L=1 TO 42
  READ h,h2,v,v2,la$
  IF hori>210 AND hori<280 AND verti>175 AND verti<
  185 THEN abbruch=1:RETURN
  IF hori>h AND hori<h2 AND verti>v AND verti<v2 T
  HEN RETURN
  NEXT
  LOCATE 4,5
  PRINT "Noch mal ";:m=MOUSE(0)
 wLm:
  IF MOUSE(0)=0 THEN GOTO wLm
  hori=MOUSE(3):verti=MOUSE(4)
  LOCATE 4,5:PRINT "           ";
  LOCATE 3,1:PRINT "                        "
  GOTO welchesLand
welcherSpieler:
  FOR p=1 TO spges
  IF Spieler$(p)=rf$(r) THEN namen$=N$(p):f=farb(p):sp
  i=p
  NEXT
    g=0:FOR p=1 TO spges
    IF rf$(p)="*" THEN g=g+1
    NEXT
    IF g=spges-1 THEN GOTO sieg
  WINDOW OUTPUT 2:LOCATE 21,28:PRINT "         "
  COLOR f,0:LOCATE 21,28:PRINT  namen$:COLOR 2,0
  GOSUB f4:CLS:B=500:GOSUB Soldat
  COLOR 1,0:PRINT namen$;" ist am Zug.";
  RETURN
wievielArmeen:
  armee=0:S=0:la=0:FOR p=1 TO 42
  IF land$(p)=rf$(r) THEN la=la+1:S=S+1
    IF p=9 THEN
      IF S=9 THEN armee=armee+5
      S=0
      ELSE
      END IF
    IF p=13 THEN
      IF S=4 THEN armee=armee+2
      S=0
      ELSE
      END IF
    IF p=20 THEN
      IF S=7 THEN armee=armee+5
      S=0
      ELSE
      END IF
    IF p=32 THEN
      IF S=12 THEN armee=armee+7
      S=0
      ELSE
      END IF
    IF p=36 THEN
      IF S=4 THEN armee=armee+2
      S=0
      ELSE
      END IF
    IF p=42 THEN
      IF la=0 THEN rf$(r)="*":GOTO ende
      IF S=6 THEN armee=armee+3
      S=0
      ELSE
      END IF
  NEXT:armeen=INT(la/3)
  IF armeen<3 THEN armeen=3
  armeen=armeen+armee:LOCATE 2,1
  PRINT namen$;" erhält ";armeen;" Armeen."
    IF rf$(r)>"F" THEN RETURN
  vt=1:LOCATE 3,1
  PRINT "Zeige mit der Maus auf eines Deiner Länder !"
  vert:
```

```
tC9         GOSUB angr1
B3          WINDOW 4,,(0,200)-(620,235),0,1
gS          GOSUB f4:Ainp=0
CD7       eing:
ju9         LOCATE 1,1
6F          INPUT"Wieviel Armeen in dieses Land";arminp
IH          Ainp=Ainp+arminp
tK          IF Ainp>armeen THEN
NcC            PRINT "Mehr als Du bekommst !"
XO             Ainp=Ainp-arminp
Wh             FOR p=1 TO 3000:NEXT:CLS
UA             GOTO eing
ud             ELSE
G9             END IF
q8A          WINDOW OUTPUT 2
09           land(L)=land(L)+arminp:GOSUB v6
op           GOSUB f4:CLS
rg           IF Ainp=armeen THEN vt=0:RETURN
NG           PRINT namen$;" Du hast noch ";armeen-Ainp;" Armeen."
Oh           armeen=armeen-Ainp
cE           PRINT "Auf welches Land !"
JT           GOTO vert
oG5     wuerfel:
WI7       LINE(400,5)-(615,42),1,B:PAINT(402,6),2,1
jv        LINE(510,5)-(510,42),1:COLOR 1,2
hq        RANDOMIZE TIMER
Dy        LOCATE 2,55:PRINT namen$:LOCATE 2,67:PRINT Znamen$
t3        w=INT(RND*7):RANDOMIZE TIMER
Xn        w2=INT(RND*7):RANDOMIZE TIMER
RB        w3=INT(RND*7)
VV9         IF w=0 OR w2=0 OR w3=0 THEN GOTO wuerfel
gw          IF w>6 OR w2>6 OR w3>6 THEN GOTO wuerfel
FI          IF w>=w2 AND w>=w3 THEN wu1(z)=w:GOTO wf1
9MA          IF w>=w2 AND w<=w3 THEN wu2(z)=w:GOTO wf2
AN           IF w>=w3 AND w<=w2 THEN wu2(z)=w:GOTO wf2
4R           IF w<=w2 AND w<=w3 THEN wu3(z)=w:GOTO wf3
5JB           wf1:
rrD             IF w2>=w3 THEN
mWF               wu2(z)=w2:wu3(z)=w3
L4                ELSE
oYH                 wu3(z)=w2:wu2(z)=w3
ibF               END IF
KXB           GOTO wurf
GV            wf2:
IpD             IF w>=w2 THEN
pYF               wu3(z)=w2:wu1(z)=w3
SB                ELSE
raH                 wu3(z)=w3:wu1(z)=w2
piF               END IF
ReD             GOTO wurf
RhB           wf3:
1zD             IF w2<=w3 THEN
saF               wu2(z)=w2:wu1(z)=w3
ZI                ELSE
ucH                 wu2(z)=w3:wu1(z)=w2
wpF               END IF
Ow6       wurf:
Oy7        IF z=1 THEN GOTO zwurf
Sw9          IF land(Anr)<3 THEN Wanz=1:GOTO wurf1
dA           IF land(Anr)<4 THEN Wanz=2:GOTO wurf2
h0           Wanz=3:GOTO wurfw
vN7        zwurf:
TJ9          IF land(Znr)<2 THEN GOTO wurf1
fS           IF land(Znr)<3 THEN GOTO wurf2
6h7        GOTO wurfw
V3         wurf1:
ey9          wu1(z)=wu2(z)
Mc           wu2(z)=0
e87        wurf2:
Sj9          wu3(z)=0
MO7        wurfw:
Zd9          IF z=1 THEN GOTO wurfw2
PEB            LOCATE 3,53:PRINT "1.";
aM             LOCATE 4,53:PRINT "2.";
1U             LOCATE 5,53:PRINT "3.";
I39          LOCATE 3,57:PRINT wu1(0):SOUND 200,.5
HA           IF wu2(0)>0 THEN LOCATE 4,57:PRINT wu2(0);:SOUND 200,.5
UL           IF wu3(0)>0 THEN LOCATE 5,57:PRINT wu3(0);:SOUND 200,.5
kC7        IF z=0 THEN z=1:GOTO wuerfel
Vp         wurfw2:
rs9          LOCATE 3,68:PRINT wu1(1)::SOUND 200,.5
dV            IF wu2(1)>0 THEN LOCATE 4,68:PRINT wu2(1);:SOUND 200,.5
qg            IF wu3(1)>0 THEN LOCATE 5,68:PRINT wu3(1);:SOUND 200,.5
9F            z=0:COLOR 1,0
pR7         RETURN
wn4      auswertung:
Y16        IF wu1(0)<=wu1(1) THEN
cv8          land(Anr)=land(Anr)-1
8r           ELSE
GNA            land(Znr)=land(Znr)-1
VO8          END IF
5z6        IF wu2(0)=0 OR wu2(1)=0 THEN GOTO aw2
10         IF wu2(0)<=wu2(1) THEN
i18          land(Anr)=land(Anr)-1
Ex           ELSE
MTA            land(Znr)=land(Znr)-1
bU8          END IF
HD6        IF wu3(0)=0 OR wu3(1)=0 THEN GOTO aw2
yF         IF wu3(0)<=wu3(1) THEN
o78          land(Anr)=land(Anr)-1
K3           ELSE
SZA            land(Znr)=land(Znr)-1
ha8          END IF
LV6        aw2:
OZ8          L=Anr:GOSUB v6
T4           f2=f:L=Znr:f=fz:GOSUB v6:f=f2:L=Anr
LW           IF land(Znr)<>0 THEN
S6A            GOSUB f4:RETURN
SB9            ELSE
oh             END IF
GA           land$(Znr)=Spieler$(spi)
kC           land(Znr)=land(Znr)+Wanz:BEEP
4U           L=Znr:GOSUB v6:land(Anr)=land(Anr)-Wanz:BEEP
yM           L=Anr:GOSUB v6:land$(Znr)=land$(Anr)
om           vs(Znr)=1
fp           WINDOW OUTPUT 4
UbB            IF Karte=0 THEN
BKD              RANDOMIZE TIMER
JM               z=INT(RND*4)
DzE                IF z=0 THEN z=1
1V                 IF z=1 THEN z$="R"
7d                 IF z=2 THEN z$="S"
mB                 IF z=3 THEN z$="K"
e7                 ka(spi)=ka(spi)+1:k$(spi,ka(spi))=z$
9r                 Karte=1
jS                 ELSE
5y                 END IF
W89            RETURN
qQ2    verschieben:
x04      CLS:LOCATE 1,1:PRINT"Verschiebe Armeen von "
1C       GOSUB welchesLand:vNr=L:vLa$=la$
T76        IF vs(vNr)=1 THEN
nt8          CLS
15           PRINT"Verschieben erst nach erneutem Angriff möglich"
oc           GOSUB pause:GOSUB pause:abbruch=1
tc           ELSE
F8           END IF
bk4      IF abbruch=1 THEN abbruch=0:CLS:RETURN
Tm       GOSUB eigLand:IF ab=1 THEN ab=0:CLS:RETURN
hw5       IF land(vNr)<2 THEN
1U7         PRINT"Hier steht nur eine Armee !":GOSUB pause
Tc          GOTO verschieben
0j          ELSE
MF          END IF
K84      LOCATE 1,1:PRINT"Verschiebe Armeen von ";vLa$;" nach "
XP       GOSUB welchesLand:nNr=L:nLa$=la$
kt       IF abbruch=1 THEN abbruch=0:CLS:RETURN
L0       LOCATE 1,1:PRINT "Verschiebe Armeen von ";vLa$;" nach ";
NQ       PRINT nLa$:GOSUB eigLand:IF ab=1 THEN ab=0:CLS:RETURN
eb8          FOR p=1 TO 8
mg           IF L=Nb(vNr,p) THEN GOTO v2
di           NEXT
da           LOCATE 3,5
Wx           PRINT "Dies ist kein Nachbarland ! Noch mal.";:GOSUB pause:GOTO verschieben
v13    v2:
BW4      GOSUB pause
7k       WINDOW 4,,(0,200)-(620,235),0,1:GOSUB f4
```

Listing 1. (Fortsetzung)

```
TF3   veing:
DO4    LOCATE 1,1
4G     INPUT "Wieviel Armeen";armeen
7t     IF armeen>=land(vNr) OR armeen<=0 THEN CLS:GOTO veing
On     land(vNr)=land(vNr)-armeen:L=vNr:GOSUB v6
kO     GOSUB f4:vs(nNr)=1
wj     land(nNr)=land(nNr)+armeen:L=nNr:GOSUB v6
fl     GOSUB f4:CLS:RETURN
RA2  eigLand:
Yw6      IF land$(L)<>rf$(r) THEN
yZ8        LOCATE 3,5:PRINT"Dies ist nicht Dein Land !"
7G         GOSUB pause:ab=1:RETURN
RA         ELSE
ng         END IF
Eq7      RETURN
Xm2  REM  *** Symbolkarten eintauschen  ***
Dx   tauschen:
Gi4    GOSUB f4:CLS:B=5
CG     t2:
Aa6      IF ka(spi)=0 THEN PRINT "Keine Karten!":RETURN
eU       IF rf$(r)>"F" THEN GOTO t3
tw       FOR D=1 TO ka(spi)
387       IF k$(spi,D)="R" THEN GOSUB Reiter
Ra        IF k$(spi,D)="S" THEN GOSUB Soldat
uW        IF k$(spi,D)="K" THEN GOSUB Kanone
YK        B=B+45:NEXT
NS6      t3:
TV7       rt=0:so=0:k=0:FOR p=1 TO ka(spi)
RL9         IF k$(spi,p)="R" THEN rt=rt+1
Oi          IF k$(spi,p)="S" THEN so=so+1
kS          IF k$(spi,p)="K" THEN k=k+1
QB7       NEXT:wk=1
vy9         IF rt>=3 OR so>=3 OR k>=3 THEN GOTO wk
4R          IF rt>=1 AND so>=1 AND k>=1 THEN wk=2:GOTO wk
YA          RETURN
KM8        wk:
rA9         IF rf$(r)>"F" THEN GOTO wKarte
pA          LOCATE 2,45
mc          PRINT"Willst Du tauschen ?";
XO          LOCATE 4,62:PRINT"Ja";:LOCATE 4,69:PRINT"Nein";
IN          LINE(480,20)-(520,35),,B:LINE(540,20)-(580,35),,B
TSA          kt:
thC            m=MOUSE(0):IF MOUSE(0)=0 THEN GOTO kt
Js             IF MOUSE(3)>480 AND MOUSE(3)<520 THEN GOTO wKarte
RS             IF MOUSE(3)>540 AND MOUSE(3)<580 THEN LOCATE 5,20:PRINT "ok";
Tt7      CLS:RETURN
hH5     wKarte:
016      IF rf$(r)<"G" THEN GOTO wk4
KN       FOR D=1 TO ka(spi)
UZ7       IF k$(spi,D)="R" THEN GOSUB Reiter
s1        IF k$(spi,D)="S" THEN GOSUB Soldat
Lx        IF k$(spi,D)="K" THEN GOSUB Kanone
zl        B=B+45:NEXT
Kg        wk4:
C69         LOCATE 1,40:PRINT namen$;" tauscht Karten gegen Soldaten !";
6R          GOSUB pause
m17       IF wk=2 THEN GOTO wk2
I19         IF rt>=3 THEN L$="R"
zo          IF so>=3 THEN L$="S"
xY          IF k>=3 THEN L$="K"
iCA          z=0:FOR p=1 TO ka(spi):IF z=3 THEN wk3
ufC            IF k$(spi,p)=L$ THEN k$(spi,p)="+":z=z+1
joB           NEXT
Ou            GOTO wk3
Nh5     wk2:
1L7       L$="R":l1$="S":l2$="K":z=0
s5        FOR p=1 TO ka(spi)
i69         IF z=3 THEN wk3
5o          IF k$(spi,p)=L$ AND rt<>0 THEN k$(spi,p)="+":z=z+1:rt=0
RA          IF k$(spi,p)=l1$ AND so<>0 THEN k$(spi,p)="+":z=z+1:so=0
TK          IF k$(spi,p)=l2$ AND k<>0 THEN k$(spi,p)="+":z=z+1:k=0
sx7       NEXT
Zu5     wk3:
C67       z=1:FOR p=1 TO ka(spi)
y89         IF k$(spi,p)<>"+" THEN K2$(1,z)=k$(spi,p):z=z+1
V98        NEXT:ka(spi)=ka(spi)-3
2F         FOR p=1 TO ka(spi)
jxA          k$(spi,p)=K2$(1,p)
z49         NEXT
pK          armeen=za:za=za+3:CLS
sd          LOCATE 1,1:PRINT namen$;" erhält ";armeen;" Armeen."
Q18        wk3.3:
vB9         IF rf$(r)<"G" THEN
BKB           PRINT "Zeige mit der Maus auf eines Deiner Länder !"
K6            GOSUB welchesLand
Sd            IF land$(L)<>rf$(r) THEN CLS:GOTO wk3.3
I8            land(L)=land(L)+armeen:GOSUB v6
x3            GOSUB f4:CLS:RETURN
fO            ELSE
1u            END IF
LS9         GOSUB wKonti:GOSUB pause:CLS
1c          PRINT  namen$;" legt seine Armeen auf ";Lname$(L);"."
cq          GOSUB pause:GOSUB v6
V77       RETURN
Bv4     Reiter:
zs      LINE(4+B,4)-(41+B,44),,B
OX      LINE(9+B,23)-(12+B,19):LINE-(11+B,18):LINE-(13+B,19)
v7      LINE-(14+B,17):LINE-(14+B,18):LINE-(16+B,22):LINE-(17+B,22)
38      LINE-(11+B,6):LINE-(13+B,7):LINE-(17+B,5):LINE-(15+B,7)
w7      LINE-(17+B,8):LINE-(13+B,9):LINE-(18+B,21):LINE-(19+B,19)
Y1      LINE-(18+B,17):LINE-(17+B,17):LINE-(18+B,14)
gw      LINE-(20+B,16):LINE-(20+B,18):LINE-(22+B,19)
6T      LINE-(23+B,24):LINE-(25+B,25):LINE-(25+B,27)
eA      LINE-(29+B,28):LINE-(32+B,28):LINE-(38+B,30)
8H      LINE-(36+B,30):LINE-(32+B,31):LINE-(30+B,30)
9I      LINE-(23+B,40):LINE-(22+B,42):LINE-(20+B,42)
FP      LINE-(21+B,41):LINE-(22+B,40):LINE-(24+B,33)
8k      LINE-(19+B,38):LINE-(19+B,39):LINE-(18+B,40)
m3      LINE-(18+B,37):LINE-(22+B,32):LINE-(20+B,32)
4P      LINE-(20+B,34):LINE-(18+B,34):LINE-(18+B,32)
Ig      LINE-(17+B,31):LINE-(14+B,35):LINE-(16+B,37)
Fc      LINE-(16+B,39):LINE-(13+B,36):LINE-(13+B,32)
DQ      LINE-(10+B,33):LINE-(8+B,37):LINE-(7+B,35):LINE-(10+B,31)
DJ      LINE-(14+B,30):LINE-(12+B,23):LINE-(10+B,25)
TA      LINE-(9+B,24):LINE-(8+B,25):LINE-(9+B,23)
TR      PAINT(18+B,28),f,1
sU      RETURN
ee2   Soldat:
MF4     LINE(4+B,4)-(41+B,44),,B
ZW      LINE(14+B,44)-(16+B,43):LINE-(17+B,36):LINE-(16+B,29)
ko      LINE-(15+B,24):LINE-(14+B,24):LINE-(14+B,22):LINE-(15+B,22)
5D      LINE-(14+B,17):LINE-(15+B,17):LINE-(13+B,10):LINE-(15+B,17)
m5      LINE-(16+B,22):LINE-(17+B,22):LINE-(17+B,17):LINE-(19+B,17)
Vj      LINE-(19+B,15):LINE-(17+B,14):LINE-(18+B,12):LINE-(17+B,12)
VU      LINE-(17+B,10):LINE-(24+B,10):LINE-(23+B,13):LINE-(23+B,16)
n6      LINE-(25+B,14):LINE-(26+B,14):LINE-(26+B,16):LINE-(27+B,17)
rC      LINE-(28+B,22):LINE-(26+B,24):LINE-(28+B,28):LINE-(26+B,30)
n8      LINE-(28+B,34):LINE-(26+B,33):LINE-(25+B,41):LINE-(26+B,44)
EP      LINE-(14+B,44):LINE(16+B,24)-(18+B,24):LINE-(18+B,27)
kG      LINE-(16+B,24):LINE(20+B,43)-(22+B,35):LINE-(23+B,43):LINE-(20+B,43)
KO      PAINT(20+B,30),f,1:RETURN
JV2   Kanone:
QL4     LINE(2+B,4)-(40+B,44),,B
KA      LINE(6+B,18)-(8+B,19):LINE-(19+B,19):LINE-(19+B,17)
iy      LINE-(23+B,17):LINE-(23+B,19):LINE-(31+B,17):LINE-(31+B,19)
km      LINE-(33+B,19):LINE-(33+B,21):LINE-(31+B,21):LINE-(31+B,22)
4Q      LINE-(29+B,22):LINE-(30+B,24):LINE-(29+B,25):LINE-(34+B,29)
Gg      LINE-(39+B,28):LINE-(36+B,30):LINE-(37+B,31):LINE-(39+B,31)
ao      LINE-(39+B,35):LINE-(10+B,35):LINE-(13+B,32):LINE-(19+B,31)
1K      LINE-(16+B,29):LINE-(15+B,27):LINE-(15+B,25):LINE-(15+B
```

```
          ,23)
1048 X5   LINE-(14+B,23):LINE-(13+B,22):LINE-(8+B,22):LINE-(6+B,2
          3)
1049 1s   LINE-(6+B,18)
1050 Ev   LINE(17+B,22)-(20+B,24):LINE-(17+B,24):LINE-(17+B,22)
1051 sm   LINE(17+B,26)-(20+B,25):LINE-(18+B,27):LINE-(17+B,26)
1052 rl   LINE(19+B,29)-(21+B,26):LINE-(21+B,30):LINE-(19+B,29)
1053 I1   LINE(23+B,30)-(23+B,26):LINE-(26+B,29):LINE-(23+B,30)
1054 42   LINE(24+B,25)-(26+B,28):LINE-(28+B,27):LINE-(24+B,25)
1055 YL   LINE(26+B,22)-(27+B,22):LINE-(27+B,24):LINE-(26+B,22)
1056 XO   PAINT(14+B,34),f,1
1057 P1   RETURN
1058 aX2 nextSpieler:
1059 Ja4   r=r+1:IF r>spges THEN r=1
1060 0J6     g=0:FOR p=1 TO 42
1061 gZ8       IF land$(p)=rf$(r) THEN g=1
1062 T1        vs(p)=0
1063 yP6     NEXT:IF g=0 AND rf$(r)>"F" THEN csp=csp-1
1064 64      IF g=0 THEN rf$(r)="*"
1065 4U4   IF rf$(r)="*" THEN GOTO nextSpieler
1066 eP    Karte=0:GOTO beginn
1067 7D0 sieg:
1068 v52  WINDOW OUTPUT 4
1069 nt   CLS
1070 kr   PRINT namen$;" ist der Sieger"
1071 0W   GOSUB pause:MENU RESET
1072 2x   END
1073 D50 f4:
1074 pm2  WINDOW OUTPUT 4:RETURN
1075 o1   ende:
1076 em4    MENU RESET:STOP
1077 lt1 REM **** Der Computer zieht ****
1078 S4  computer:
1079 PM3   IF za>21 THEN dugmax=10
1080 Nz    IF za>33  THEN dugmax=15
1081 yi    GOSUB wKonti
1082 cv5     PRINT  namen$;" legt seine Armeen auf ";Lname$(L)
1083 283   GOSUB v6:GOSUB pause
1084 xG5     dug=dug+(.8/csp):angriff=0:angri=0
1085 Bg3   comangr:
1086 sZ7       ON MENU GOSUB ende:MENU ON
1087 I6        IF dug>dugmax THEN dug=dugmax
1088 ji        IF ka(spi)>2 THEN GOSUB tauschen
1089 Ju        maxi=-100:max=-100:angriff=0:anz=0
1090 Ch        FOR L=1 TO 42
1091 Ao9         IF land$(L)=rf$(r) AND land(L)>1 THEN GOSUB ang
1092 hm7       NEXT
1093 up6      IF maxi<2 THEN angriff=0:GOSUB cver:GOTO ang2
1094 mI       IF angriff=1 THEN GOTO ang1
1095 nJ       IF angriff=0 THEN GOTO ang2
1096 E6       ang:
1097 EW8        FOR N=1 TO 10
1098 8N         Nbar=Nb(L,N):IF Nbar=0 THEN RETURN
1099 aJ         IF land$(Nbar)<>rf$(r) THEN max=land(L)-land(Nbar
            ):angriff=1
1100 nx         IF land$(Nbar)<"G" THEN max=max+.5
1101 RI         IF max>maxi THEN
1102 5dC            maxi=max:Anr=L:Znr=Nbar
1103 1F             FOR p=1 TO spges
1104 wjE              IF land$(Znr)=Spieler$(p) THEN fz=farb(p)
1105 UrC            NEXT p
1106 RA             ELSE
1107 ng             END IF
1108 x29          NEXT
1109 EH6      ang1:
1110 dT8        AusgL$=Lname$(Anr):ZielL$=Lname$(Znr)
1111 9N         FOR p=1 TO spges
1112 kIA          IF land$(Znr)=Spieler$(p) THEN Znamen$=N$(p)
1113 278        NEXT
1114 PQ         GOSUB f4:CLS
1115 3j         PRINT "Von ";AusgL$;" nach ";ZielL$
1116 Zu         GOSUB pause
1117 dI         z=0:GOSUB wuerfel:GOSUB pause:GOSUB auswertung
1118 Vm        angri=angri+.5:GOSUB pause
1119 TX6      ang2:
1120 vZ8        IF angri>=dug OR angriff=0 THEN
1121 OsC            angriff=0:anz=0:GOTO nextSpieler
1122 hQ             ELSE
1123 3w             END IF
1124 os9          GOTO comangr
1125 Za3   pause:
1126 jB5     FOR p=1 TO 3000:NEXT:RETURN
1127 Jd0 REM  *** Computer verschiebt Armeen ***
1128 6K2  cver:
1129 pK4    FOR L=1 TO 42
1130 zF6      IF land$(L)=rf$(r) AND land(L)>3 THEN GOSUB cverp
1131 KP4    NEXT
1132 cE6      RETURN
1133 FP3   cverp:
1134 4C5     S=0:FOR p=1 TO 8:N(p)=Nb(L,p)
1135 w99         IF N(p)=0 THEN GOTO cv1
1136 nM7       IF land$(N(p))<>rf$(r) THEN S=1
1137 r15     cv1:
1138 qK      NEXT:IF S=1 THEN angriff=0:RETURN
1139 F1      armeen=land(L)-2
1140 f16      n2=0:FOR p=1 TO 8:FOR D=1 TO 8
1141 kr8        n1=Nb(N(p),D)
1142 fR9          IF n1=0 THEN GOTO cv2
1143 018        IF land$(n1)<>rf$(r) THEN S=1:n2=N(p)
1144 2D6      cv2:
1145 Pz       NEXT:NEXT:IF n2=0 THEN angriff=0:RETURN
1146 PP5     GOSUB f4:CLS:PRINT namen$;" verschiebt von ";
1147 vp6      PRINT Lname$(L);" nach ";Lname$(n2);" ";
1148 fZ       PRINT armeen;" Armeen."
1149 6R8        GOSUB pause
1150 sk         land(L)=land(L)-armeen:GOSUB v6
1151 LI         land(n2)=land(n2)+armeen:L=n2:GOSUB v6
1152 S6         GOSUB pause:angriff=1
1153 xZ6      RETURN
1154 dR3   wKonti:
1155 205     conti=0:contimax=16:FOR L=1 TO 9
1156 3g      IF land$(L)=rf$(r) THEN conti=conti+1
1157 kp      NEXT
1158 4H2 IF conti=0 THEN GOTO Samerika
1159 RF9       IF land$(27)=rf$(r) THEN conti=conti+.5
1160 8t        IF land$(10)=rf$(r) THEN conti=conti+.5
1161 H6        IF land$(14)=rf$(r) THEN conti=conti+.5
1162 p5        IF conti=10.5 THEN contimax=15:conti=0:GOTO Sameri
           ka
1163 Y1        contimax=(10.5/conti):conti=0:l1=1:l2=L-1
1164 J64   Samerika:
1165 k45     FOR L=10 TO 13
1166 Dq6      IF land$(L)=rf$(r) THEN conti=conti+1
1167 uz5     NEXT
1168 g07       IF conti=0 THEN GOTO Europa
1169 dm9         IF land$(9)=rf$(r) THEN conti=conti+.5
1170 iS          IF land$(37)=rf$(r) THEN conti=conti+.5
1171 C1          contimax2=5/conti
1172 Ho          GOSUB conti2
1173 SS          IF contimax2<contimax THEN contimax=contimax2:l1=
             10:l2=L-1
1174 VU          conti=0
1175 1t4   Europa:
1176 0M6     FOR L=14 TO 20
1177 017      IF land$(L)=rf$(r) THEN conti=conti+1
1178 5A6     NEXT
1179 dH8       IF conti=0 THEN GOTO Asien
1180 ElA         IF land$(3)=rf$(r) THEN conti=conti+.5
1181 bJ          IF land$(21)=rf$(r) THEN conti=conti+.5
1182 eN          IF land$(22)=rf$(r) THEN conti=conti+.5
1183 hR          IF land$(23)=rf$(r) THEN conti=conti+.5
1184 wg          IF land$(37)=rf$(r) THEN conti=conti+.5
1185 zk          IF land$(38)=rf$(r) THEN conti=conti+.5
1186 d0          contimax2=10/conti:GOSUB conti2
1187 1R          IF contimax2<contimax THEN contimax=contimax2:l1
             =14:l2=20
1188 ji          conti=0
1189 vT4   Asien:
1190 Jg6     FOR L=21 TO 32
1191 cF7      IF land$(L)=rf$(r) THEN conti=conti+1
1192 J06     NEXT
1193 WV8       IF conti=0 THEN GOTO Austra
1194 ysA         IF land$(19)=rf$(r) THEN conti=conti+.5
1195 xq          IF land$(18)=rf$(r) THEN conti=conti+.5
1196 Av          IF land$(38)=rf$(r) THEN conti=conti+.5
1197 Dz          IF land$(39)=rf$(r) THEN conti=conti+.5
1198 21          IF land$(33)=rf$(r) THEN conti=conti+.5
1199 5m          IF land$(34)=rf$(r) THEN conti=conti+.5
1200 M1          IF land$(1)=rf$(r) THEN conti=conti+.5
1201 01          contimax2=15.5/conti:GOSUB conti2
1202 Ok          IF contimax2<contimax THEN contimax=contimax2:l1
             =21:l2=32
1203 yx          conti=0
1204 1n5       Austra:
```

Listing 1. (Fortsetzung)

```
3X7      FOR L=33 TO 36
rU9        IF land$(L)=rf$(r) THEN conti=conti+1
Yd7      NEXT
Qk9        IF conti=0 THEN GOTO Afrika
9nB          IF land$(31)=rf$(r) THEN conti=conti+.5
pw           contimax2=4.5/conti:GOSUB conti2
tH           IF contimax2<contimax THEN contimax=contimax2:l
             1=33:12=36
76           conti=0
ob5     Afrika:
Ch7      FOR L=37 TO 42
0d9        IF land$(L)=rf$(r) THEN conti=conti+1
hm7      NEXT
ZK9        IF conti=0 THEN GOTO kontiend
8vB          IF land$(12)=rf$(r) THEN conti=conti+.5
JB           IF land$(17)=rf$(r) THEN conti=conti+.5
MF           IF land$(18)=rf$(r) THEN conti=conti+.5
J3           IF land$(23)=rf$(r) THEN conti=conti+.5
kR           contimax2=8/conti:GOSUB conti2
3S           IF contimax2<contimax THEN contimax=contimax2:l
             1=37:12=42
JI           conti=0
Md5     kontiend:
TG7      max=0:maxi=0:start=0
vU       FOR L=11 TO 12:p=0
he9        IF land$(L)<>rf$(r) THEN GOTO p2
fN7      p:
Fm9        p=p+1:nbl=Nb(L,p)
pe         IF nbl=0 THEN GOTO p2
tz         IF land$(nbl)=rf$(r) THEN GOTO p
3t         IF land$(nbl)<>rf$(r) AND nbl>11-1 AND nbl<12+
           1 THEN  max=max+3:IF max>maxi THEN maxi=max:13=L:
           start=1:GOTO p
1M         IF land$(nbl)<>rf$(r) THEN max=max+1:IF max>max
           i THEN maxi=max:13=L:start=1
BS         GOTO p
447      p2:
9b9        max=0:NEXT
Ux6     IF start=1 THEN start=0:GOTO gefunden
9d8       FOR L=1 TO 42:p=0
wu        IF land$(L)<>rf$(r) THEN GOTO p3
IL7      p5:
Ry8       p=p+1:nbl=Nb(L,p)
3t        IF nbl=0 THEN GOTO p3
9k        IF land$(nbl)<>rf$(r) THEN land(13)=land(13)+arme
          en:GOTO p4
St        IF land$(nbl)=rf$(r) THEN GOTO p5
HI7      p3:
CH9        NEXT
MO7      p4:
mG9        L=13:start=0:RETURN
Nz6     gefunden:
FN8       land(13)=land(13)+armeen:L=13
YA        RETURN
us6     conti2:
yN8       IF contimax2=1 THEN contimax2=17
bD        RETURN
v92   mausposition:
aZ4     DATA 15,70,0,40,"Alaska"
Kw      DATA 78,175,0,35,"Nord-West-Territorium"
mG      DATA 180,290,0,33,"Grönland"
H1      DATA 75,160,37,54,"Alberta"
kQ      DATA 163,205,36,55,"Ontario"
LO      DATA 212,245,41,59,"Quebeck"
mD      DATA 71,141,57,78,"Weststaaten"
rw      DATA 143,216,57,89,"Oststaaten"
dP      DATA 70,135,79,104,"Mittelamerika"
rs      DATA 64,175,104,122,"Venezuela"
Py      DATA 56,116,123,162,"Peru"
Gs      DATA 107,213,123,167,"Brasilien"
Vb      DATA 71,136,160,220,"Argentinien"
Oq      DATA 250,303,43,61,"Island"
UO      DATA 305,352,36,73,"Skandinavien"
Mr      DATA 290,360,77,91,"Mitteleuropa"
5N      DATA 266,312,89,109,"Westeuropa"
1H      DATA 316,367,91,108,"SÜdeuropa"
3y      DATA 355,424,18,100,"Ukraine"
ZC      DATA 234,280,74,96,"Groß Britannien"
aN      DATA 430,493,4,70,"Ural"
VG      DATA 427,478,71,110,"Afghanistan"
zs      DATA 377,446,104,126,"Mittlerer Osten"
hb      DATA 496,533,5,78,"Sibirien"
YB      DATA 540,587,5,28,"Jakutien"
uL      DATA 532,560,29,75,"Irkutsk"
GM      DATA 575,615,0,79,"Kamtschatka"
rn      DATA 517,561,75,91,"Mongolei"
Bk      DATA 479,566,85,110,"China"
Jv      DATA 449,511,112,141,"Indien"
cG      DATA 513,560,116,140,"Siam"
xk      DATA 569,600,80,110,"Japan"
NX      DATA 575,620,135,160,"Neu Guinea"
Sq      DATA 510,575,141,163,"Indonesien"
oS      DATA 518,570,163,200,"West-Australien"
GU      DATA 575,620,160,195,"Ost-Australien"
cN      DATA 245,300,110,156,"Nord-west-Afrika"
h8      DATA 305,380,112,131,"Ägypten"
qW      DATA 356,405,132,167,"Ostafrika"
05      DATA 320,357,146,164,"Kongo"
Oc      DATA 305,380,168,210,"Südafrika"
ZO      DATA 390,420,169,200,"Madagaskar"
FR      REM gespeichert in  Nb
KE2   Nachbarland:
h04     DATA 2,4,27,0,0,0,0,0
MW      DATA 1,3,5,4,0,0,0,0
Kk      DATA 2,5,6,14,0,0,0,0
Tf      DATA 1,2,5,7,0,0,0,0
Mi      DATA 2,3,6,4,7,8,0,0
ig      DATA 3,5,8,0,0,0,0,0
CK      DATA 4,5,8,9,0,0,0,0
4S      DATA 9,7,5,6,0,0,0,0
vi      DATA 7,8,10,0,0,0,0,0
nj      DATA 11,12,9,0,0,0,0,0
bh      DATA 10,12,13,0,0,0,0,0
OP      DATA 10,11,13,37,0,0,0,0
PC      DATA 11,12,0,0,0,0,0,0
eh      DATA 15,20,3,0,0,0,0,0
g3      DATA 14,20,16,19,0,0,0,0
tj      DATA 15,20,19,18,17,0,0,0
J6      DATA 20,16,18,37,0,0,0,0
UW      DATA 17,16,19,37,38,23,0,0
TJ      DATA 15,16,18,23,22,21,0,0
66      DATA 14,15,16,17,0,0,0,0
VG      DATA 22,29,24,19,0,0,0,0
JI      DATA 21,29,30,23,19,0,0,0
6k      DATA 22,30,19,18,38,39,0,0
ek      DATA 21,29,28,26,25,0,0,0
pq      DATA 24,26,27,0,0,0,0,0
Xd      DATA 25,24,28,27,0,0,0,0
ka      DATA 25,26,28,32,1,0,0,0
Ni      DATA 27,26,24,29,32,0,0,0
fS      DATA 28,24,21,22,30,31,32,0
HK      DATA 23,22,29,31,0,0,0,0
ue      DATA 30,29,34,0,0,0,0,0
PN      DATA 27,28,29,0,0,0,0,0
U8      DATA 34,36,0,0,0,0,0,0
bS      DATA 33,35,36,31,0,0,0,0
WA      DATA 34,36,0,0,0,0,0,0
pt      DATA 33,35,34,0,0,0,0,0
dW      DATA 40,39,38,17,18,12,0,0
2s      DATA 37,39,23,18,0,0,0,0
Fx      DATA 38,37,40,41,42,23,0,0
KA      DATA 37,39,41,0,0,0,0,0
6r      DATA 40,39,42,0,0,0,0,0
TX      DATA 39,41,0,0,0,0,0,0
(C) 1987 M&T
```

Listing 1. (Schluß)

Programmname:	Lader
Computer:	A 500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2
Bemerkung:	Lädt Listing 1 nach, das unter dem Namen »Hauptprogramm« gespeichert sein muß

```
Programm : Lader
----------------

 NJO REM Basic-Speicher vergrößern
 Dk  CLEAR,40000&
 03  CHAIN "Hauptprogramm"
(C) 1987 M&T
```

Listing 2. Das Ladeprogramm erweitert den Basic-Speicher. Bitte mit dem Checksummer (Seite 159) eingeben.

Fortsetzung von Seite 22

Finanzen im Griff

```
20 y4  farbe0=0
21 5D  farbe1=1
22 CM  farbe2=2
23 JV  farbe3=3
24 Qe  farbe4=4
25 Xn  farbe5=5
26 ew  farbe6=6
27 15  farbe7=7
28 xF  WINDOW OUTPUT 2
29 HO  COLOR farbe6,0
30 2D  LOCATE 1,1
31 sh  PRINT "Zeit";
32 8K  LOCATE 2,1
33 SP  PRINT "Datum";
34 ER  LOCATE 3,1
35 g3  PRINT "Bezugszeit";
36 KY  LOCATE 4,1
37 4D  PRINT "Kontoname";
38 sB  LOCATE 1,30
39 01  PRINT "Arbeitsdauer";
40 yI  LOCATE 2,30
41 sL  PRINT "Buchungen";
42 4P  LOCATE 3,30
43 gi  PRINT "Speicher";
44 7T  WINDOW OUTPUT 1
45 81  ON ERROR GOTO fehler
46 Od  ON BREAK GOSUB unterbrechung
47 jw  BREAK ON
48 mr  ON MENU GOSUB wahl
49 oM  ON TIMER(1) GOSUB uhr
50 RX  TIMER ON
51 3T  arbeitsdauer=TIMER
52 1a  GOSUB aktuell
53 4N  GOSUB pruefung
54 6W  MENU ON
55 2Y  BEEP
56 DO  WHILE 1
57 iL  SLEEP
58 NB  WEND
59 CM  wahl:
60 Ks  v=MENU(0):w=MENU(1)
61 HB  MENU OFF
62 Tk  ON v GOSUB haupt1,haupt2,haupt3,haupt4,haupt5,haupt6,haupt7
63 Cl  GOSUB aktuell
64 Fa  weiter:
65 Hh  MENU ON
66 Q2  RETURN
67 QZ  haupt1:
68 ru  konto$=verzeichnis$(w):kontonr=w
69 T5  RETURN
70 Xi  haupt3:
71 Rt  ON w GOTO menu31,menu32,menu33
72 Xh  haupt2:
73 6Q  ON w GOTO menu21,menu22,menu23,menu25,menu26,menu27,menu28,
       menu29
74 dp  haupt4:
75 Nj  ON w GOTO menu41,menu42
76 hu  haupt5:
77 zN  ON w GOTO menu51,menu52,menu53
78 lz  haupt6:
79 Fb  ON w GOTO menu61,menu62,menu63
80 p4  haupt7:
81 ZJ  ON w GOTO menu71,menu72,menu73,menu74,menu75,menu76
82 a6  menu21:
83 yv  IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
       RN
84 E4  IF index=99 THEN CALL meldung("Buchungen nur nach Neustart
       möglich"):RETURN
85 dJ  CALL requester ("Um welche Art Buchung handelt es sich","Ei
       nnahme","Ausgabe")
86 60  IF req THEN
87 LZ  vorzeichen=1
88 1k  ELSE
89 Hi  vorzeichen=-1
90 OH  END IF
91 MK  ein:
92 rV  WINDOW 5,,(0,100)-(maxbreite,135),0,1
93 nS  LOCATE 2,2:COLOR farbe5,0:PRINT "Bezeichnung"
94 00  LOCATE 4,2:PRINT "Betrag";
95 uR  LOCATE 2,40:COLOR farbe5,0:PRINT  STRING$(20,"_");STRING$(2
       0," ")
96 ZI  LOCATE 2,40:COLOR farbe1,0:INPUT a$:IF LEN(a$)>18 THEN BEE
       P:GOTO ein
97 Be  IF a$="" THEN menu211
98 GP  ein1:
99 My  LOCATE 4,40:COLOR farbe5,0:PRINT  STRING$(14,"_");STRING$(2
       0," ")
100 8J  LOCATE 4,40:COLOR farbe1,0:INPUT b$
101 ce  IF VAL(b$)>999999999.99# THEN BEEP:GOTO ein1
102 OM  IF VAL(b$)=0 THEN  menu211
103 Om  index=index+1
104 kf  nummer(index)=kontonr
105 e6  konto$(index)=konto$
106 p5  jahr$(index)=jahr$
107 4Z  monat$(index)=monat$
108 En  tag$(index)=tag$
109 gv  buchung$(index)=UCASE$(a$)
110 Tb  betrag$(index)=STR$(VAL(b$)*vorzeichen)
101 ce  IF VAL(b$)>999999999.99# THEN BEEP:GOTO ein1
102 OM  IF VAL(b$)=0 THEN  menu211
103 Om  index=index+1
104 kf  nummer(index)=kontonr
105 e6  konto$(index)=konto$
106 p5  jahr$(index)=jahr$
107 4Z  monat$(index)=monat$
108 En  tag$(index)=tag$
109 gv  buchung$(index)=UCASE$(a$)
110 Tb  betrag$(index)=STR$(VAL(b$)*vorzeichen)
111 I5  menu211:
112 Fy  WINDOW CLOSE 5
113 Ea  WINDOW OUTPUT 1
114 Co  RETURN
115 9g  menu22:
116 Bn  WINDOW 5,,(0,100)-(maxbreite,115),0,1
117 Ux  LOCATE 2,2:COLOR farbe5,0:PRINT "Bitte neuen Kontonamen ein
        geben"
118 20  LOCATE 2,40:COLOR farbe5,0:PRINT  STRING$(20,"_")
119 R9  LOCATE 2,40:COLOR farbe1,0:INPUT a$:IF LEN(a$)>18 THEN BEE
        P:GOTO menu22:
120 e3  IF a$="" THEN menu221
121 fb  a$=UCASE$(a$)
122 Rf  CALL requester ("Kontonamen "+a$+" speichern ?","Ja","Nein"
        )
123 hz  IF req THEN
124 k04     konto$=a$
125 DQ      maxkonto=maxkonto+1
126 K7      verzeichnis$(maxkonto)=a$
127 fD      flag%(maxkonto)=1
128 N2      Z=0
129 Zv      IF maxkonto<>0 THEN
130 mp      FOR i=1 TO maxkonto
131 jK      IF flag%(i)=0 THEN Z=i
132 Zp      NEXT i
133 vf      IF Z<>0 THEN
134 OJ      SWAP verzeichnis$(maxkonto),verzeichnis$(Z)
135 9H      SWAP flag%(maxkonto),flag%(Z)
136 W1      maxkonto=maxkonto-1
137 92      END IF
138 A3      END IF
139 NW      OPEN "O",#1,verzeichnis$+jahr$
140 wz      FOR i=1 TO maxkonto
141 L4      PRINT #1,verzeichnis$(i)
142 mi      PRINT #1,flag%(i)
143 zX      MENU 1,i,flag%(i),verzeichnis$(i)
144 11      NEXT i
145 ZM      CLOSE #1
146 Q1      GOSUB einlesen
147 JCO END IF
148 vj  menu221:
149 qZ  WINDOW CLOSE 5
150 pB  WINDOW OUTPUT 1
151 nP  RETURN
152 mK  menu23:
153 63  IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETU
        RN
154 8y  CALL requester("Soll Konto "+konto$+" gelöscht werden ?","J
        a","Nein")
155 DV  IF req THEN
156 en  OPEN "O",#1,verzeichnis$+jahr$
157 DG  FOR i=1 TO maxkonto
158 jF  IF konto$<>verzeichnis$(i) THEN
```

Listing 1. (Fortsetzung)

```
dM PRINT #1,verzeichnis$(i)
40 PRINT #1,flag%(i)
Cv ELSE
nx PRINT #1,verzeichnis$(i):flag%(i)=0
kr PRINT #1,0
sS MENU 1,i,0,verzeichnis$(i)
bU END IF
7N NEXT i
vi CLOSE #1
DO OPEN "R",#2,jahr$,8
UT FIELD #2,8 AS x$
wG LSET x$=MKD$(0)
c2 PUT #2,kontonr
4s CLOSE #2
UM OPEN "R",#2,jahr$+monat$,8
ZY FIELD #2,8 AS x$
1L LSET x$=MKD$(0)
h7 PUT #2,kontonr
9x CLOSE #2
Oz konto$=""
pi END IF
Gs RETURN
Jt menu25:
7d summe#=0
aX IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETURN
A8 OPEN "I",#1,konto$+jahr$+monat$
ex zeile=4
RR WHILE NOT EOF(1)
PU IF zeile=4 THEN
ag CLS
Wz LINE (0,10)-(520,10),farbe4
Zp LINE (0,25)-(520,25),farbe4
UB LINE (520,0)-(520,200),farbe4
Pb LINE (0,0)-(0,200),farbe4
yi COLOR farbe7,0
7H LOCATE 1,2:PRINT "Einzelkonto";TAB(45);"Jahr";TAB(55);"Monat";
kH LOCATE 3,2:PRINT "Tag"
m9 LOCATE 3,12:PRINT "Monat"
9P LOCATE 3,22:PRINT "Jahr"
fq LOCATE 3,32:PRINT "Bezeichnung"
QJ LOCATE 3,52:PRINT "Betrag"
21 COLOR farbe6,0
Tz LOCATE 1,15
qE PRINT konto$;TAB(50);jahr$;TAB(60)monat$;
wc COLOR farbe3,0
E7 END IF
uI zeile=zeile+1
x3 INPUT #1,a$,b$,c$,d$,e
d7 LOCATE zeile,2
Ob PRINT c$;
Rb LOCATE zeile,13:PRINT b$;
Nb LOCATE zeile,22:PRINT a$;
fx LOCATE zeile,32:PRINT d$;
qt LOCATE zeile,51:PRINT USING "##########.##";e;
cd LOCATE zeile+1,1:PRINT PTAB(1);SPACE$(63);
5M summe#=summe#+e:LOCATE zeile+2,45
uS LINE (1,(zeile+1)*8)-(519,(zeile+1)*8),farbe0
Qk LINE (1,(zeile+2)*8)-(519,(zeile+2)*8),farbe4
Ox LINE (1,(zeile)*8-1)-(519,(zeile)*8-1),farbe0
Iz LINE (1,(zeile+1)*8-1)-(519,(zeile+1)*8-1),farbe4
Om PRINT USING "Saldo:##########.##";summe#
W7 IF zeile=23 THEN zeile=4:CALL meldung ("Maustaste")
Oo WEND
ob CLOSE #1
zX CALL meldung("Maustaste")
Bp COLOR farbe1,0
BH CLS
Oc RETURN
5g menu26:
rN summe#=0
MD OPEN "R",#1,jahr$+monat$,8
pS FIELD #1,8 AS ein$
HN CLS
dg LINE (0,10)-(380,10),farbe4
gW LINE (0,25)-(380,25),farbe4
TS LINE (380,0)-(380,200),farbe4
6I LINE (0,0)-(0,200),farbe4
fP COLOR farbe7,0
oT LOCATE 1,2:PRINT "Monatskonto";TAB(30);"Jahr";TAB(40);"Monat";
Wk LOCATE 3,2:PRINT "Konto"
qG LOCATE 3,31:PRINT "Betrag"
gP COLOR farbe6,0
Jh LOCATE 1,35
tP PRINT jahr$;TAB(46)monat$;
aG COLOR farbe3,0
cf FOR i=1 TO maxkonto
VH LSET ein$=""
V1 GET #1,i
cy LOCATE i+4,2
wQ PRINT verzeichnis$(i),
3J LOCATE i+4,30:PRINT USING "##########.##";CVD(ein$);
yL LOCATE i+5,1:PRINT PTAB(1);SPACE$(45);
GT summe#=summe#+CVD(ein$):LOCATE i+6,24
eP LINE (1,(i+5)*8)-(379,(i+5)*8),farbe0
wn LINE (1,(i+6)*8)-(379,(i+6)*8),farbe4
F6 LINE (1,(i+4)*8-1)-(379,(i+4)*8-1),farbe0
VS LINE (1,(i+5)*8-1)-(379,(i+5)*8-1),farbe4
zN PRINT USING "Saldo:##########.##";summe#
aq NEXT i
OB CLOSE #1
Z7 CALL meldung("Maustaste")
1P COLOR farbe1,0
1r CLS
aC RETURN
hJ menu27:
Rx summe#=0
kW OPEN "R",#1,jahr$,8
P2 FIELD #1,8 AS ein$
rx CLS
DG LINE (0,10)-(380,10),farbe4
G6 LINE (0,25)-(380,25),farbe4
32 LINE (380,0)-(380,200),farbe4
gs LINE (0,0)-(0,200),farbe4
Fz COLOR farbe7,0
sX LOCATE 1,2:PRINT "Jahreskonto";TAB(30);"Jahr";
6K LOCATE 3,2:PRINT "Konto"
Qq LOCATE 3,31:PRINT "Betrag"
Gz COLOR farbe6,0
tH LOCATE 1,35
n2 PRINT jahr$;
Aq COLOR farbe3,0
CF FOR i=1 TO maxkonto
5r LSET ein$=""
5b GET #1,i
CY LOCATE i+4,2
Fy PRINT verzeichnis$(i);
dt LOCATE i+4,30:PRINT USING "##########.##";CVD(ein$);
Yv LOCATE i+5,1:PRINT PTAB(1);SPACE$(45);
q3 summe#=summe#+CVD(ein$):LOCATE i+6,24
Ez LINE (1,(i+5)*8)-(379,(i+5)*8),farbe0
WN LINE (1,(i+6)*8)-(379,(i+6)*8),farbe4
pg LINE (1,(i+4)*8-1)-(379,(i+4)*8-1),farbe0
52 LINE (1,(i+5)*8-1)-(379,(i+5)*8-1),farbe4
Zx PRINT USING "Saldo:##########.##";summe#
AQ NEXT i
yl CLOSE #1
9h CALL meldung("Maustaste")
Lz COLOR farbe1,0
LR CLS
Am RETURN
Jw menu28:
3I IF index=0 THEN CALL meldung("Buchungsspeicher leer"):RETURN
su zaehler=0
au zeile=5
Dz WHILE zaehler<index
TH zaehler=zaehler+1
PQ IF zeile=5 THEN
Ua CLS
K5 LINE (0,18)-(570,18),farbe4
ko LINE (0,33)-(570,33),farbe4
sj LINE (570,0)-(570,200),farbe4
JV LINE (0,0)-(0,200),farbe4
sc COLOR farbe7,0
oj LOCATE 2,22:PRINT "Aktuelle Buchungen:"
Pu LOCATE 4,2:PRINT "Konto";
sb LOCATE 4,22:PRINT "Tag";
rg LOCATE 4,26:PRINT "Monat";
ER LOCATE 4,32:PRINT "Jahr"
q8 LOCATE 4,38:PRINT "Bezeichnung"
d4 LOCATE 4,59:PRINT "Betrag"
uc COLOR farbe5,0
6z END IF
mA zeile=zeile+1
1P LOCATE zeile,2:PRINT konto$(zaehler);
```

```
eF  LOCATE zeile,22:PRINT tag$(zaehler);
wk  LOCATE zeile,27:PRINT monat$(zaehler);
mv  LOCATE zeile,32:PRINT jahr$(zaehler);
i9  LOCATE zeile,38:PRINT buchung$(zaehler);
j9  LOCATE zeile,59:PRINT USING "##########.##";VAL(betrag$(zaehler));
Lx  IF zeile=23 THEN zeile=5:CALL meldung ("Maustaste")
kY  WEND
iG  CALL meldung("Maustaste")
uY  COLOR farbe1,0
u0  CLS
jL  RETURN
uY  menu29:
cr  IF index=0 THEN CALL meldung("Buchungsspeicher leer"):RETURN
ic  GOSUB schreiben
nP  RETURN
oG  menu31:
17  CLS
lr  CHDIR"df0:"
al  COLOR farbe7,farbe0
Um  FILES
MP  CALL meldung ("Maustaste")
LQ  COLOR farbe1,farbe0
7D  CLS
Ju  CHDIR"df0:Schublade"
xZ  RETURN
OT  menu32:
BH  CLS
v1  CHDIR"df0:"
kv  COLOR farbe7,farbe0
ew  FILES
Jt  COLOR farbe6,farbe0
j3  INPUT "Bitte Filenamen eingeben ",a$
HN  CLS
Xc  COLOR farbe1,farbe0
oh  KILL a$
V6  CHDIR"df0:Schublade"
91  RETURN
Ei  menu33:
v0  CHDIR "df0:Schublade"
uF  GOSUB einlesen
OW  BEEP
Eq  RETURN
Lj  menu41:
y6  GOSUB systemzeit
FG  GOTO einlesen
Qp  menu42:
ji  WINDOW 5,,(0,100)-(maxbreite,147),16,1
1w  LINE (80,5)-(130,50),farbe3,bf
ZE  LINE (235,5)-(285,50),farbe3,bf
Rr  LINE (390,5)-(440,50),farbe3,bf
DP  LINE (540,17)-(610,37),farbe3,bf
JW  COLOR farbe6,farbe3
Nt  LOCATE 4,12
XN  PRINT USING "####";jahr
k5  LOCATE 4,33:PRINT USING "##";monat
qq  LOCATE 4,52:PRINT USING "##";tag
z0  LOCATE 4,70:PRINT "fertig";
v0  COLOR farbe1,farbe0
VT  LOCATE 4,5:PRINT "Jahr";
qc  LOCATE 4,23:PRINT "Monat";
Hv  LOCATE 4,45:PRINT "Tag";
Tg  COLOR farbe6,farbe3
wc  menu421:
qi  WHILE MOUSE(0)<>0:WEND
Do  WHILE MOUSE(0)=0:WEND
xY  x=MOUSE(1):y=MOUSE(2)
yI  IF x>540 AND x<610 AND y>17 AND y<37 THEN menu422
fD  IF y<5 AND y>50 THEN menu421
Xh  IF y<27 THEN
oB2   IF x>80 AND x<130 THEN jahr=jahr+1
9p    IF x>235 AND x<285 THEN monat=monat+1:IF monat>12 THEN monat=1
eX    IF x>390 AND x<440 THEN tag=tag+1:IF tag>maxtag THEN tag=1
yh0 ELSE
OP2   IF x>80 AND x<130 THEN jahr=jahr-1
bJ    IF x>235 AND x<285 THEN monat=monat-1:IF monat<1 THEN monat=12
xW    IF x>390 AND x<440 THEN tag=tag-1:IF tag<1 THEN tag=maxtag
NG0 END IF
Ma  jahr$=RIGHT$(STR$(jahr),LEN(STR$(jahr))-1)
YY  monat$=RIGHT$(STR$(monat),LEN(STR$(monat))-1):IF LEN(monat$)=1 THEN monat$="0"+monat$
s0  tag$=RIGHT$(STR$(tag),LEN(STR$(tag))-1):IF LEN(tag$)=1 THEN tag$="0"+tag$
G8  IF jahr$+monat$<"198801" THEN jahr=1988:monat=1
Uw  IF jahr$>RIGHT$(DATE$,4) THEN jahr=VAL(RIGHT$(DATE$,4)):monat=12
eE  maxtag=28+(monat\7+2)*(1-monat MOD 2)+(3-monat\9)*(monat MOD 2)+(monat=2)*2+(jahr MOD 4=0)*(monat=2)
cj  IF tag>maxtag THEN tag=maxtag
qo  LOCATE 4,12:PRINT USING "####";jahr;
rS  LOCATE 4,33:PRINT USING "##";monat;
5W  LOCATE 4,52:PRINT USING "##";tag;
La  GOTO menu421
P6  menu422:
Ym  jahr$=RIGHT$(STR$(jahr),LEN(STR$(jahr))-1)
kk  monat$=RIGHT$(STR$(monat),LEN(STR$(monat))-1):IF LEN(monat$)=1 THEN monat$="0"+monat$
4C  tag$=RIGHT$(STR$(tag),LEN(STR$(tag))-1):IF LEN(tag$)=1 THEN tag$="0"+tag$
8r  WINDOW CLOSE 5
7T  WINDOW OUTPUT 1
n8  GOSUB einlesen
6i  RETURN
Jd  menu51:
KQ  CLS
9W  FOR i=0 TO 7
ge  LINE (i*70+10,20)-(i*70+50,50),1,b:REM Farbfelder
hy  PAINT (i*70+15,30),i,1
q8  LOCATE 8,1
dK  PRINT PTAB(i*70+20);i
JZ  NEXT i
gQ  LOCATE 2,2:PRINT "Farben"
ES  LOCATE 12,2:PRINT "rot"
tI  LOCATE 14,2:PRINT "grün"
Xs  LOCATE 16,2:PRINT "blau"
4M  FOR i=0 TO 2
Ya  LINE (100,86+i*16)-(400,96+i*16),1,b:REM Einzelfarben
Qg  NEXT i
mZ  LINE (130,145)-(220,165),1,b:PAINT (180,155),farbe3,1
Ip  farbe=0:COLOR farbe6,farbe3
BD  LOCATE 20,20:PRINT "fertig";
Js  xpos(1)=rot(farbe+1)*291+101
zt  xpos(2)=gruen(farbe+1)*291+101
cS  xpos(3)=blau(farbe+1)*291+101
Jd  FOR i=1 TO 3
1N  LOCATE 10+i*2,1
ju  PRINT PTAB(xpos(i));"]";
aq  NEXT i
PJ  schritt=.05
yZ  menu511:
Cs  LINE (farbe*70+8,18)-(farbe*70+52,52),1,b
Cs  LINE (farbe*70+7,17)-(farbe*70+53,53),1,b
U3  xpos(1)=rot(farbe+1)*291+101
A4  xpos(2)=gruen(farbe+1)*291+101
nd  xpos(3)=blau(farbe+1)*291+101
2K  COLOR 1,0
Vp  FOR i=1 TO 3
DZ  LOCATE 10+i*2,1
v6  PRINT PTAB(xpos(i));"]";
m2  NEXT i
yq  WHILE MOUSE(0)<>0:WEND
Lw  WHILE MOUSE(0)=0:WEND
oc  maus=MOUSE(0):xmouse=MOUSE(1):ymouse=MOUSE(2)
Ub  IF ymouse>145 AND ymouse<165 AND xmouse>130 AND xmouse<220 THEN menu512
am  IF ymouse>20 AND ymouse<50 THEN
nAR                     FOR i=0 TO 7
XC                      IF xmouse>i*70+10 AND xmouse<i*70+50 THEN
BSU                       COLOR 0,0
n3                        FOR j=1 TO 3
Ro                        LOCATE 10+j*2,1
AM                        PRINT PTAB(xpos(j));"]";
OH                        NEXT j
T8                        LINE (farbe*70+8,18)-(farbe*70+52,52),0,b
T8                        LINE (farbe*70+7,17)-(farbe*70+53,53),0,b
kV                        farbe=i
XQR                     END IF
```

Listing 1. (Fortsetzung)

```
3J                           NEXT i
Qa                           GOTO menu511
aTO  END IF
t7   IF ymouse>86 AND ymouse<96 THEN
OTU                             COLOR 0,0:LOCATE 12,1
iN                              IF xmouse<100 OR xmouse<xpos(1) THEN
dxX                               rot(farbe+1)=rot(farbe+1)-schritt
fYV                              END IF
Dp                               IF xmouse>400 OR xmouse>xpos(1) THEN
cuX                               rot(farbe+1)=rot(farbe+1)+schritt
ibV                              END IF
gxO  PRINT PTAB(xpos(1));"] ";
kd   END IF
Xg   IF ymouse>102 AND ymouse<112 THEN
KpU                             COLOR 0,0:LOCATE 14,1
yZ                              IF xmouse<100 OR xmouse<xpos(2) THEN
VhX                               gruen(farbe+1)=gruen(farbe+1)-schritt
piV                              END IF
T1                               IF xmouse>400 OR xmouse>xpos(2) THEN
MgX                               gruen(farbe+1)=gruen(farbe+1)+schritt
slV                              END IF
tBO  PRINT PTAB(xpos(2));"] ";
un   END IF
Zw   IF ymouse>118 AND ymouse<128 THEN
eBU                             COLOR 0,0:LOCATE 16,1
El                              IF xmouse<100 OR xmouse<xpos(3) THEN
CeX                               blau(farbe+1)=blau(farbe+1)-schritt
zsV                              END IF
jD                               IF xmouse>400 OR xmouse>xpos(3) THEN
7XX                               blau(farbe+1)=blau(farbe+1)+schritt
2vV                              END IF
6PO  PRINT PTAB(xpos(3));"] ";
4x   END IF
Hw   IF rot(farbe+1)>1 THEN rot(farbe+1)=1
8j   IF rot(farbe+1)<0 THEN rot(farbe+1)=0
7q   IF gruen(farbe+1)>1 THEN gruen(farbe+1)=1
tY   IF gruen(farbe+1)<0 THEN gruen(farbe+1)=0
WY   IF blau(farbe+1)>1 THEN blau(farbe+1)=1
IG   IF blau(farbe+1)<0 THEN blau(farbe+1)=0
5b   PALETTE farbe,rot(farbe+1),gruen(farbe+1),blau(farbe+1)
3D   GOTO menu511
7j   menu512:
WF   CALL requester ("Farben abspeichern","Ja","Nein")
xm   IF req=1 THEN
Nc   OPEN "O",#1,"Farben"
nC   FOR i=1 TO 8
kx   PRINT #1,rot(i)
sJ   PRINT #1,gruen(i)
QW   PRINT #1,blau(i)
q6   NEXT i
eR   CLOSE #1
NG   END IF
06   CLS
pR   RETURN
4P   menu52:
Ok   GOTO farben
8U   menu53:
fK   IF index<>0 THEN
1D   BEEP:BEEP:BEEP
kC   CALL meldung("Buchungsspeicher entleeren !!!")
wY   RETURN
Cv   ELSE
oC   MENU RESET
pV   WINDOW CLOSE 2
1Q   WINDOW CLOSE 1
RW   SCREEN CLOSE 1
bL   SYSTEM
dW   END IF
Mc   menu61:
3R   sum#=0
NK   IF konto$="" THEN CALL meldung("Kein Konto definiert"):RETURN
xv   OPEN "I",#1,konto$+jahr$+monat$
Zr   OPEN "lpt1:" FOR OUTPUT AS #3
Eo   PRINT #3, "Einzelkonto: ";konto$;TAB(40);"Jahr ";jahr$;TAB(55);"Monat ";monat$
Vg   PRINT #3,
Jr   PRINT #3, "Tag";TAB(10);"Bezeichnung";TAB(40);"Betrag"
fh   PRINT #3, STRING$(52,"_")
II   WHILE NOT EOF(1)
c0   INPUT #1,a$,b$
9u   INPUT #1,a$,b$,c$
LS   PRINT #3,a$;TAB(10);b$;TAB(40);
fp   PRINT #3,USING "#########.##";VAL(c$)
gE   sum#=sum#+VAL(c$)
OC   WEND
np   PRINT #3, STRING$(52,"_")
Ph   PRINT #3,TAB(29);
Jh   PRINT #3,USING "     Saldo:#########.##";sum#
F2   CLOSE #1
OD   CLOSE #3
P1   RETURN
k1   menu62:
Pn   sum#=0
Gr   IF maxkonto=0 THEN RETURN
md   OPEN "R",#1,jahr$+monat$,8
vD   OPEN "lpt1:" FOR OUTPUT AS #3
c4   FIELD #1,8 AS a$
Ra   PRINT #3, "Monatskonto:";TAB(40);"Jahr ";jahr$;TAB(55);"Monat ";monat$
s3   PRINT #3,
V1   PRINT #3,"Konto";TAB(30);"Betrag"
xy   PRINT #3, STRING$(42,"_")
ux   FOR i=1 TO maxkonto
hN   PRINT #3,verzeichnis$(i);TAB(30);
81   LSET a$="":GET #1,i
km   PRINT #3,USING "#########.##";CVD(a$)
jJ   sum#=sum#+CVD(a$)
k0   NEXT i
45   PRINT #3, STRING$(42,"_")
5p   PRINT #3,TAB(22);
r7   PRINT #3,USING " Saldo:#########.##";sum#
4F   PRINT #3,
cP   CLOSE #1
1a   CLOSE #3
m0   RETURN
9R   menu63:
mA   sum#=0
dE   IF maxkonto=0 THEN RETURN
xj   OPEN "R",#1,jahr$,8
Ia   OPEN "lpt1:" FOR OUTPUT AS #3
zR   FIELD #1,8 AS a$
Jc   PRINT #3,"Jahreskonto:";TAB(40);"Jahr ";jahr$;TAB(55);"Monat ";monat$
FQ   PRINT #3,
s8   PRINT #3,"Konto";TAB(30);"Betrag"
KL   PRINT #3, STRING$(42,"_")
HK   FOR i=1 TO maxkonto
4k   PRINT #3,verzeichnis$(i);TAB(30);
V8   LSET a$="":GET #1,i
79   PRINT #3,USING "#########.##";CVD(a$)
6g   sum#=sum#+CVD(a$)
7N   NEXT i
RS   PRINT #3, STRING$(42,"_")
SC   PRINT #3,TAB(22);
ms   PRINT #3,USING " Saldo:#########.##";sum#
y1   CLOSE #1
7w   CLOSE #3
8k   RETURN
Xj   menu71:
U8   drucker CHR$(27)+CHR$(64)+CHR$(27)+"l"+CHR$(10)+CHR$(27)+"N"+CHR$(4)
Bn   RETURN
cp   menu72:
mU   drucker CHR$(12)
Eq   RETURN
hv   menu73:
pY   drucker CHR$(27)+"!"+CHR$(0)+CHR$(27)+"x1"
Ht   RETURN
m1   menu74:
98   drucker CHR$(27)+"x0"+CHR$(27)+"!"+CHR$(0)
Kw   RETURN
r7   menu75:
EE   drucker CHR$(27)+"x0"+CHR$(27)+"!"+CHR$(1)
Nz   RETURN
aW   unterbrechung:
Fd   MENU RESET
Ad   CHDIR "df0:Schublade"
```

```
pk END
3y fehler:
Gm BEEP
ki IF ERR=53 THEN CALL meldung("File existiert nicht"):RESUME weiter
gz nummer$=STR$(ERR)
PL CALL meldung("Fehler "+nummer$)
Nl MENU RESET
I1 CHDIR "df0:Schublade"
xs END
yQ einlesen:
Fp RESTORE einlesen
QU DATA "Verzeichnis"
UQ DATA "Option","Buchen              ","neues Konto       ","Konto löschen      "
Yz DATA          "Übersicht Einzelkonto","Übersicht Monat      ","Übersicht Jahr       ","Buchungen              ","Eintragen          ","#"
dn DATA "Disk","Directory      ","File löschen","Diskwechsel ","#"
2I DATA "Zeit","Systemzeit","Bezugszeit","#"
61 DATA "System","Farben    ","Farbreset","Ende       ","#"
Tk DATA "Ausgabe","Einzelkonto","Monatskonto","Jahreskonto","#"
uD DATA "Drucker","RESET    ","Vorschub","NLQ      ","DRAFT    ","ELITE    ","$"
ay MENU RESET
vW konto$=""
ya READ verzeichnis$:MENU 1,0,1,verzeichnis$
kk OPEN "A",#1,verzeichnis$+jahr$:CLOSE #1
IL OPEN "I",#1,verzeichnis$+jahr$
2B maxkonto=0
vv WHILE NOT EOF(1)
hu maxkonto=maxkonto+1
Sf INPUT #1,verzeichnis$(maxkonto)
Ek INPUT #1,flag%(maxkonto)
UE MENU 1,maxkonto,flag%(maxkonto),verzeichnis$(maxkonto)
Oo WEND
ob CLOSE #1
Un i=2:j=0
9E einles:
Mz READ a$
aG IF a$="#" THEN i=i+1:j=0:GOTO einles
Wa IF a$<>"$" THEN MENU i,j,1,a$:j=j+1:GOTO einles
2e RETURN
2R bildschirm:
QD SCREEN 1,640,256,3,2
fu maxbreite=631:maxhoehe=242
mp WINDOW 2,,(0,215)-(maxbreite,maxhoehe),16,1:COLOR 1,0
cF WINDOW 1,"von  A.N. u. M.F."+SPACE$(10)+"+++Haushaltsverwaltung+++"+SPACE$(20)+"c 1988",(0,0)-(maxbreite,200),16,1
8k RETURN
G3 farben:
AI RESTORE farben
Fb DATA 0,0,0
Sm DATA 1,1,1
In DATA .4,.6,1
XJ DATA 1,.73,0
XM DATA .33,.87,0
FB DATA .73,1,0
6Y DATA .93,.2,0
g0 DATA 1,.6,.67
Jg FOR i=0 TO 7
5h READ rot(i+1),gruen(i+1),blau(i+1)
BN PALETTE i,rot(i+1),gruen(i+1),blau(i+1)
Rh NEXT i
Nz RETURN
yy systemzeit:
gz jahr$=RIGHT$(DATE$,4):jahr=VAL(jahr$)
az monat$=LEFT$(DATE$,2):monat=VAL(monat$)
Xp tag$=MID$(DATE$,4,2):tag=VAL(tag$)
Tp datum$=tag$+"-"+monat$+"-"+jahr$
T5 RETURN
HB SUB meldung(a$) STATIC
98 SHARED maxbreite
43 a=LEN(a$)*8
Cy WINDOW 3,,(maxbreite-a-60,0)-(maxbreite,15),16,1
kq CLS
Da COLOR 6,0
bg LOCATE 2,(a+60)/8/2-LEN(a$)/2
M1 PRINT a$:COLOR 1,0
sk WHILE MOUSE(0)<>0:WEND
qv WHILE MOUSE(0)=0
jX WEND
i4 WINDOW OUTPUT 1
bI WINDOW CLOSE 3
NP END SUB
qH SUB requester(a$,b$,c$) STATIC
Y1 SHARED req
6A a=LEN(a$)*8+60:b=LEN(b$)*8+60:c=LEN(c$)*8+60
m7 IF a>b+c THEN
bOD          maxleng=a
219       ELSE
wjD          maxleng=b+c
PIO END IF
8R WINDOW 5,,(0,0)-(maxleng,40),16,1
KG LOCATE 2,1:COLOR 6,0
ea PRINT PTAB((maxleng-LEN(a$)*8)/2);a$;
1o LINE (20,30)-(b-20,40),1,b:PAINT (22,35),2,1
v8 LINE (maxleng-c+20,30)-(maxleng-20,40),1,b:PAINT (maxleng-c+22,35),2,1
Rk COLOR 0,2
Od LOCATE 5,1
CR PRINT PTAB(30);b$;
Qf LOCATE 5,1
Sy PRINT PTAB(maxleng+30-c);c$;
ZG reques1:
H9 WHILE MOUSE(0)<>0:WEND
eF WHILE MOUSE(0)=0:WEND
Uf IF MOUSE(2)<30 OR MOUSE(2)>40 THEN reques1
UN IF MOUSE(1)>20 AND MOUSE(1)<b-20 THEN req=1:GOTO reques2
zs IF MOUSE(1)>maxleng-c+20 AND MOUSE(1)<maxleng-20 THEN req=0:GOTO reques2
fv GOTO reques1
jR reques2:
Dw WINDOW CLOSE 5
CY WINDOW OUTPUT 1
qs END SUB
Ri uhr:
OV MENU STOP
Ef fenster=WINDOW(1)
Nf WINDOW OUTPUT 2
7d LOCATE 1,15
fN COLOR farbe5,0
mJ PRINT TIME$
Sm LOCATE 1,45
D5 COLOR farbe2,farbe0:zeit=INT(TIMER)
9z stunde=(zeit-arbeitsdauer)\3600
In minute=(zeit-arbeitsdauer-stunde*3600)\60
3E sekunde=INT(zeit-arbeitsdauer-stunde*3600-minute*60+.5)
Aa PRINT USING "h:##";stunde;
M2 PRINT USING " m:##";minute;
vg PRINT USING " s:##";sekunde;
i4 LOCATE 3,45
Nf PRINT USING "######";FRE(-1);FRE(-2);FRE(0);
5x WINDOW OUTPUT fenster
L1 MENU ON
U6 RETURN
d2 aktuell:
fx WINDOW OUTPUT 2
we COLOR farbe5,0
U1 LOCATE 2,15
jg PRINT datum$
a8 LOCATE 3,15
Xk PRINT tag$+"-"+monat$+"-"+jahr$;
gF LOCATE 4,15
ks PRINT SPACE$(20);
iH LOCATE 4,15
I1 PRINT konto$;
uF LOCATE 2,45
PP PRINT USING "##";index;
17 WINDOW OUTPUT 1
jL RETURN
3R schreiben:
U0 FOR zaehler=1 TO index
Qz OPEN "A",#1,konto$(zaehler)+jahr$(zaehler)+monat$(zaehler)
ZT PRINT #1,jahr$(zaehler)
I3 PRINT #1,monat$(zaehler)
H9 PRINT #1,tag$(zaehler)
QK PRINT #1,buchung$(zaehler)
Kx PRINT #1,betrag$(zaehler)
kX CLOSE #1
tW OPEN "R",#1,jahr$(zaehler),8
gL OPEN "R",#2,jahr$(zaehler)+monat$(zaehler),8
2U FIELD #1,8 AS a$
Dc FIELD #2,8 AS b$
48 GET #1,nummer(zaehler):a#=CVD(a$)
jX LSET a$=MKD$(VAL(betrag$(zaehler))+a#)
```

Listing 1. (Fortsetzung)

```
PUT #1,nummer(zaehler)
GET #2,nummer(zaehler):b#=CVD(b$)
LSET b$=""
LSET b$=MKD$(VAL(betrag$(zaehler))+b#)
PUT #2,nummer(zaehler)
CLOSE #1
CLOSE #2
WINDOW OUTPUT 2
COLOR farbe5,0
LOCATE 2,45
PRINT USING "##";index-zaehler;
WINDOW OUTPUT 1
NEXT zaehler
index=0
CLS
RETURN
SUB neudat(file$) STATIC
OPEN "R",#1,file$,8
FIELD #1,8 AS a$
LSET a$=MKD$(0)
FOR i=1 TO 19
PUT #1,i
NEXT i
CLOSE #1
OPEN "O",#1,"zeit"
PRINT #1,RIGHT$(DATE$,4)+LEFT$(DATE$,2)
CLOSE #1
END SUB
pruefung:
OPEN "I",#1,"Zeit"
INPUT #1,zeit$
CLOSE #1
IF LEFT$(zeit$,4)<RIGHT$(DATE$,4) THEN
        CALL neudat(RIGHT$(DATE$,4))
        FOR j=1 TO 12
        j$=STR$(j):j$=RIGHT$(j$,LEN(j$)-1)
        IF LEN(j$)=1 THEN j$="0"+j$
        CALL neudat(RIGHT$(DATE$,4)+j$)
        NEXT j
        CALL meldung ("Jahresdatei geschrieben")
END IF
RETURN
SUB drucker(a$) STATIC
OPEN "par:" FOR OUTPUT AS #3
PRINT #3,a$;
CLOSE #3
END SUB
(C) 1987 M&T
```

Listing 1. (Schluß)

Fortsetzung von Seite 27

Intelligenter Sprachentrainer

```
      READ t$(i)
    IF t$(i)<>"-1" THEN NEXT
    FOR i=1 TO 10
      READ ff(i)
    NEXT
    FOR i=1 TO 16
      READ f!(i)
    NEXT
    anwenderladen
REM  Bildschirmaufbau
    SCREEN 1,640,244,3,2
    WINDOW 1,"Vokabel - Trainer  Vers. 1.0
          von  Andreas Regul  (c) 7/1988",(0,0)-(631,230),
          0,1
    FOR i=1 TO 4
      MENU i,0,0,""
    NEXT
    FOR i=0 TO 7
      FOR j=0 TO 2
        READ farb(i,j)
      NEXT
      PALETTE i,f!(farb(i,0)),f!(farb(i,1)),f!(farb(i,2))
    NEXT
    COLOR 3,2
    e$=DATE$
    dat$=MID$(e$,4,2)+"."+LEFT$(e$,2)+"."+RIGHT$(e$,4)
    fenster 0,0,632,23
    funktion 1
    pixlocate 348,16:PRINT "Datum: " dat$
    pixlocate 502,16:PRINT "Zeit: " TIME$
    ON TIMER (1) GOSUB uhr
    TIMER ON
REM  Anwendernamen eingeben
  anweingabe:
    fenster 0,30,320,48
    TIMER OFF
    pixlocate 16,49:PRINT "Bitte den Anwendernamen eingeb
    en:"
    TIMER ON
    kasten 16,60,181,11
    z$=""
    stringeingabe 21,68,20,5
    z$=UCASE$(z$)
    FOR i=1 TO man
    IF na$(i)<>z$ THEN NEXT:GOSUB neuanwender ELSE anw=
    i:GOSUB anwbenutzung
    FOR i=0 TO 7
      FOR j=0 TO 2
        farb(i,j)=fa(anw,i,j)
      NEXT
    NEXT
    FOR i=0 TO 7
      PALETTE i,f!(farb(i,0)),f!(farb(i,1)),f!(farb(i,2))
    NEXT
    mabn=mabn+1
    IF mabn>10 THEN mabn=10:FOR i=0 TO 9:FOR j=0 TO 2:ab
    n$(i,j)=abn$(i+1,j):NEXT j,i
    abn$(mabn,0)=dat$
    abn$(mabn,1)=LEFT$(TIME$,5)
    datnamenlad
REM  Hauptmenue
  hauptmenu:
    loeschen
    funktion 1
    menue 0,30,210,2,10
    IF be THEN hauptmenu
    ON aw GOSUB diskette,eingabe,ausgabe,editieren,suche,
    abfrage,statistik,ende,sonderfunktion
    GOTO hauptmenu
REM  Diskettenbetrieb
  diskette:
    funktion 2
    menue 100,60,220,11,14
    IF be THEN RETURN
    ON aw GOSUB laden,speichern,datloeschen,wechseln
    RETURN
REM  Vokabeldatei laden
  laden:
    loeschen
    funktion 11
    IF md=0 THEN fehler 0,30,31:RETURN
    IF va THEN frage 0,30,35:IF aw=0 THEN RETURN
    dateiauswahl
    IF be THEN RETURN
    e$=d$(aw)+"."+LEFT$(na$(anw),3)
    lad=aw
    OPEN e$ FOR APPEND AS #1
    e=LOF(1)
    CLOSE 1
    IF e=0 THEN KILL e$:fehler 50,58,34:RETURN
    OPEN e$ FOR INPUT AS #1
    INPUT #1,mbn
    FOR i=1 TO mbn
      FOR j=0 TO 2
        INPUT #1,bn$(i,j)
      NEXT
    NEXT
    INPUT #1,mv
    FOR i=1 TO mv
      FOR j=0 TO 1
        LINE INPUT #1,v$(i,j)
      NEXT
```

```
109 70          FOR j=0 TO 5
110 CG8           INPUT #1,st(i,j)
111 sx6         NEXT
112 ty4       NEXT
113 dN        CLOSE 1
114 v5        va=0
115 Dp        RETURN
116 3WO   REM  Vokabeldatei speichern
117 1n2     speichern:
118 Vf4       loeschen
119 wO        funktion 12
120 we        IF mv=0 THEN fehler 0,30,32:RETURN
121 7S        fenster 0,30,344,57
122 9j        TIMER OFF
123 OM        COLOR 3,2
124 sY        pixlocate 16,48:PRINT "Dateiname    :"
125 yq        pixlocate 16,62:PRINT "Dateilänge   : " mv "Vokabeln"
126 3X        pixlocate 16,78:PRINT "Erste Eingabe:"
127 OJ        kasten 136,40,178,11
128 yD        pixlocate 143,78:IF lad THEN PRINT da$(lad) ELSE PRIN
              T dat$
129 6S        COLOR 3,2
130 Jp        TIMER ON
131 RB        z$=d$(lad)
132 uR        stringeingabe 140,48,20,5
133 eB        IF z$="" THEN RETURN
134 UE        z$=UCASE$(z$)
135 Ow        IF z$=d$(lad) THEN datspeichern
136 bx        FOR i=1 TO md
137 Tg6         IF UCASE$(d$(i))=z$ THEN fehler 100,60,36:RETURN
138 J04       NEXT
139 FV        IF md>99 THEN fehler 100,80,38:RETURN
140 Aa        st=0
141 8t2     datspeichern:
142 nP4       e$=z$+"."+LEFT$(na$(anw),3)
143 6H        OPEN e$ FOR OUTPUT AS #1
144 Fk        PRINT #1,mbn
145 uU        FOR i=1 TO mbn
146 Zn6         FOR j=0 TO 2
147 QV8           PRINT #1,bn$(i,j)
148 TY6         NEXT
149 UZ4       NEXT
150 Dm        PRINT #1,mv
151 Oe        FOR i=1 TO mv
152 cp6         FOR j=0 TO 1
153 OX8           PRINT #1,v$(i,j)
154 Ze6         NEXT
155 r8          FOR j=0 TO 5
156 LM8           PRINT #1,st(i,j)
157 ch6         NEXT
158 di4       NEXT
159 N7        CLOSE 1
160 4k        IF st THEN RETURN
161 gq        va=0
162 PY        IF z$=d$(lad) THEN RETURN
163 Me        FOR i=md TO 0 STEP -1
164 Oi        IF d$(i)>z$ THEN NEXT
165 Kv        FOR j=md+1 TO i+2 STEP -1
166 X36         d$(j)=d$(j-1)
167 j1          da$(j)=da$(j-1)
168 ns4       NEXT
169 6W        d$(i+1)=z$:da$(i+1)=dat$
170 Ig        md=md+1:lad=i+1
171 Ok        datnamensp
172 8k        RETURN
173 t90   REM  Vokabeldatei loeschen
174 A12     datloeschen:
175 Qa4       loeschen
176 tM        funktion 13
177 4v        IF md=0 THEN fehler 0,30,31:RETURN
178 80        dateiauswahl
179 1C        n=aw
180 81        IF n=lad THEN lad=0
181 DJ        IF lad>n THEN lad=lad-1
182 by        IF be THEN RETURN
183 Fn        frage 40,80,37
184 mT        IF aw=0 THEN RETURN
185 Rb        e$=d$(n)+"."+LEFT$(na$(anw),3)
186 B8        KILL e$
187 Ni        FOR i=n TO md
188 eB6         d$(i)=d$(i+1)
189 o7          da$(i)=da$(i+1)
190 9E4       NEXT
191 lu        md=md-1
192 j5        datnamensp
193 T5        RETURN
194 ZUO   REM  Diskette wechseln
195 Uu2     wechseln:
196 lv4       loeschen
197 Gk        funktion 14
198 Ec        IF va AND lad THEN frage 0,30,39:IF aw=0 THEN RETURN
199 oy        loeschen
200 RC        fehler 0,30,40
201 kc        CHDIR "DFO:"
202 4N        datnamenlad
203 q2        IF be THEN fehler 0,30,41:RETURN
204 zL        fenster 0,30,632,190
205 U4        TIMER OFF
206 c4        pixlocate 16,47:PRINT "Vokabeldateien"
207 3D        pixlocate 250,47:PRINT "Erstellungsdatum"
208 z5        TIMER ON
209 cB        y=54
210 n9        FOR i=1 TO md
211 T96         y=y+10
212 om          IF y>210 THEN y=204:SCROLL (8,55)-(340,204),0,-10
213 cC          TIMER OFF
214 Iq          pixlocate 16,y:PRINT d$(i)
215 kZ          pixlocate 250,y:PRINT da$(i)
216 7D          TIMER ON
217 af4       NEXT
218 03        WHILE MOUSE(0)<0:WEND
219 V6        WHILE MOUSE(0)=0:WEND
220 uW        RETURN
221 BPO   REM  Vokabeln eingeben
222 Xd2     eingabe:
223 OZ4       WHILE INKEY$<>"":WEND
224 DN        loeschen
225 GI        funktion 3
226 rB        vokfenster
227 fD        franzfont 0,116
228 h5        fenster 260,116,372,83
229 sS        TIMER OFF
230 J5        COLOR 3,2
231 TI        pixlocate 272,133:PRINT "Information:"
232 bq        pixlocate 272,149:PRINT "Vokabelanzahl :"
233 Lz        pixlocate 272,163:PRINT "Speicherbedarf:"
234 7h        kasten 400,141,51,11
235 WB        kasten 400,155,51,11
236 hO        kasten 560,141,51,11
237 5U        COLOR 3,5
238 9H        pixlocate 406,149:PRINT USING "###";mv
239 o2        pixlocate 406,163:PRINT USING "### %";mv*100/mmv
240 tF        COLOR 3,2
241 19        pixlocate 462,149:PRINT "von maximal"
242 AZ        COLOR 3,5
243 Nt        pixlocate 566,149:PRINT USING "####";mmv
244 Zf        TIMER ON
245 IJ        kasten 272,173,339,11
246 SK        LINE (272,175)-(272+mv*336/mmv,181),3,bf
247 2n2     vokeingabe:
248 ET4       mv=mv+1
249 19        IF mv>mmv THEN mv=mmv:fehler 100,100,33:RETURN
250 Dn        TIMER OFF
251 J1        COLOR 3,5
252 oe        pixlocate 16,61:PRINT SPACE$(LEN(v$(mv-1,0)))
253 tH        pixlocate 16,99:PRINT SPACE$(LEN(v$(mv-1,1)))
254 jp        TIMER ON
255 RK        z$=""
256 rs        stringeingabe 16,61,73,5
257 8a        IF z$="" THEN mv=mv-1:RETURN
258 7C        v$(mv,0)=z$:z$="":va=1
259 kw        stringeingabe 16,99,73,5
260 Bd        IF z$="" THEN mv=mv-1:RETURN
261 Co        v$(mv,1)=z$
262 Pz        TIMER OFF
263 Vu        COLOR 3,5
```

Listing 2. Das Hauptprogramm »Trainer HP« muß sich auf der Diskette im Laufwerk befinden, das im Ladeprogramm angegeben ist. Ansonsten erscheint »File not fond«. Bitte mit dem Checksummer (Seite 159) eingeben.

```
264 Zh      pixlocate 406,149:PRINT USING "###";mv
265 V3      pixlocate 406,163:PRINT USING "###";mv*100/mmv
266 qU      LINE (272+(mv-1)*336/mmv,175)-(272+mv*336/mmv,181),3,
            bf
267 w2      TIMER ON
268 DX      GOTO vokeingabe
269 Cx0 REM  Vokabeln ausgeben
270 502   ausgabe:
271 y84     loeschen
272 75      funktion 4
273 RA      IF mv=0 THEN fehler 0,30,42:RETURN
274 Wj      fenster 0,30,350,75
275 FN      fenster 24,116,350,75
276 YV      information 380,30
277 8R      fenster 396,116,236,75
278 Sq      kasten 408,126,205,11
279 Hd      kasten 408,142,205,11
280 qJ      kasten 408,158,205,11
281 f6      kasten 408,174,205,11
282 jJ      TIMER OFF
283 pE      COLOR 3,5
284 hl      pixlocate 418,134:PRINT "1   10   Vokabel zurück"
285 Wq      pixlocate 418,150:PRINT "1   10   Vokabel weiter"
286 Ea      pixlocate 418,166:PRINT "Statistik ausgeben"
287 1A      pixlocate 418,182:PRINT "Zum Hauptmenü"
288 HN      TIMER ON
289 Fo      LINE (438,127)-(439,135),3,b
290 S5      LINE (476,127)-(477,135),3,b
291 xS      LINE (438,143)-(439,151),3,b
292 AJ      LINE (476,143)-(477,151),3,b
293 N1      n=1:n1=n
294 ON      z$=v$(n,0)
295 DJ      vokausgabe 11,46,z$
296 7V      z$=v$(n,1)
297 n6      vokausgabe 35,132,z$
298 j92   abfrage7:
299 hM4     WHILE MOUSE(0)<0:WEND
300 oP      WHILE MOUSE(0)=0:WEND
301 uP      x=MOUSE(3):y=MOUSE(4)
302 cH      IF y<120 THEN ausgrafik
303 dV      IF x<399 OR x>626 OR y>191 THEN abfrage7
304 Vk      IF y>172 THEN RETURN
305 WY      IF y>156 THEN vokstatist
306 VH      IF y>140 THEN IF x>439 AND x<480 THEN n=n+10:GOTO
            vokausg ELSE n=n+1:GOTO vokausg
307 MB      IF y>119 THEN IF x>439 AND x<480 THEN n=n-10 ELSE
            n=n-1
308 f02   vokausg:
309 JN4     IF n<1 THEN n=mv ELSE IF n>mv THEN n=1
310 GJ      LINE (11,38)-(338,102),2,bf
311 AZ      LINE (35,124)-(360,189),2,bf
312 If      z$=v$(n,0)
313 Vb      vokausgabe 11,46,z$
314 Pn      z$=v$(n,1)
315 50      vokausgabe 35,132,z$
316 Hr      TIMER OFF
317 Nm      COLOR 3,5
318 F7      pixlocate 572,78:PRINT USING "####";n
319 ms      TIMER ON
320 GT      LINE (n1*216/mmv+393,90)-(n1*216/mmv+393,96),3
321 c6      LINE (n*216/mmv+393,90)-(n*216/mmv+393,96),4
322 hx      n1=n
323 ih      GOTO abfrage7
324 D12   ausgrafik:
325 qo4     IF x<396 OR x>611 OR y<89 OR y>99 THEN abfrage7
326 qr      x=x-394
327 4C      n=CINT(x*mmv/216)
328 jR      IF n>mv THEN n=n1:GOTO abfrage7
329 Ou      GOTO vokausg
330 7p2   vokstatist:
331 TL4     fenster 0,195,632,35
332 X7      TIMER OFF
333 Ok      COLOR 3,2
334 XP      pixlocate 8,210:PRINT "Übersetzung abfragen:"
335 xj      pixlocate 8,224:PRINT "Fremdwort abfragen  :";
336 aI      pixlocate 346,210:PRINT "Schrift üben:";
337 gb      kasten 180,202,161,11
338 55      kasten 180,216,161,11
339 Fm      kasten 456,202,161,11
340 k9      COLOR 3,5
341 UX      pixlocate 184,210:PRINT USING "### von ";st(n,1);
342 z9      PRINT USING "### = ";st(n,0);
343 OS      IF st(n,0) THEN PRINT USING "### %";st(n,1)*100/st
            (n,0) ELSE PRINT "  0 %"
344 r1      pixlocate 184,224:PRINT USING "### von ";st(n,3);
345 8K      PRINT USING "### = ";st(n,2);
346 r0      IF st(n,2) THEN PRINT USING "### %";st(n,3)*100/st
            (n,2); ELSE PRINT "  0 %";
347 Wa      pixlocate 460,210:PRINT USING "### von ";st(n,5);
348 HV      PRINT USING "### = ";st(n,4);
349 Ka      IF st(n,4) THEN PRINT USING "### %";st(n,5)*100/st
            (n,4) ELSE PRINT "  0 %"
350 MO      LINE (580,216)-(616,226),5,bf
351 9o      rahmen 580,216,37,11
352 6C      pixlocate 582,224:PRINT " OK";
353 KQ      TIMER ON
354 Vt2   abfrage5:
355 bG4     WHILE MOUSE(0)<0:WEND
356 iJ      WHILE MOUSE(0)=0:WEND
357 oJ      x=MOUSE(3):y=MOUSE(4)
358 UZ      IF x<580 OR x>616 OR y<216 OR y>226 THEN abfrage5
359 kE      LINE (0,195)-(631,230),1,bf
360 JI      GOTO abfrage7
361 Vy0 REM  Vokabeln editieren
362 MW2   editieren:
363 ga4     funktion 5
364 Un      IF mv=0 THEN fehler 100,70,43:RETURN
365 kN      menue 100,70,200,44,46
366 Zw      IF be THEN RETURN
367 ue      ON aw GOTO verbessern,vokloeschen,sploeschen
368 Bn0 REM  Vokabeln verbessern
369 VR2   verbessern:
370 Zj4     loeschen
371 Me      funktion 44
372 9d      vokauswahl
373 g3      IF be THEN RETURN
374 dn      loeschen
375 Ga      vokfenster
376 ZW      franzfont 0,120
377 Gq      TIMER OFF
378 M1      COLOR 3,5
379 ac      pixlocate 16,99:PRINT v$(aw,1)
380 lr      TIMER ON
381 Y7      WHILE INKEY$<>"":WEND
382 HH      z$=v$(aw,0)
383 uv      stringeingabe 16,61,73,5
384 pi      IF z$<>"" THEN v$(aw,0)=z$:va=1
385 QM      z$=v$(aw,1)
386 nz      stringeingabe 16,99,73,5
387 WA      IF z$<>"" THEN v$(aw,1)=z$
388 cE      RETURN
389 W10 REM  Vokabeln loeschen
390 KF2   vokloeschen:
391 u44     loeschen
392 j2      funktion 45
393 Uy      vokauswahl
394 10      IF be THEN RETURN
395 Vg      n=aw
396 hL      frage 50,70,47
397 Du      IF aw=0 THEN RETURN
398 wZ      FOR i=n TO mv
399 bo6       FOR j=0 TO 1
400 r68         v$(i,j)=v$(i+1,j)
401 Yd6       NEXT
402 q7        FOR j=0 TO 5
403 BS8         st(i,j)=st(i+1,j)
404 bg6       NEXT
405 ch4     NEXT
406 q7      mv=mv-1
407 ju      va=1
408 wY      RETURN
409 Ab0 REM  Speicher loeschen
410 d02   sploeschen:
411 E04     loeschen
412 5P      funktion 46
413 56      IF va THEN frage 0,30,48:IF aw=0 THEN RETURN
414 vC      IF va=0 THEN frage 0,30,49:IF aw=0 THEN RETURN
415 Gu      FOR i=1 TO mv
416 4L6       FOR j=0 TO 5
417 1J8         st(i,j)=0
418 pu6       NEXT
419 qv4     NEXT
420 0L      mv=0:va=0:lad=0:mbn=0
421 91      RETURN
```

```
r40  REM Vokabeln suchen
GB2    suche:
1b4      funktion 6
jM       menue 100,84,200,15,16
Xu       IF be THEN RETURN
vd       ON aw GOTO spsuche,disksuche
1a0  REM Vokabeln im Speicher suchen
VD2    spsuche:
Xh4      loeschen
vd       IF mv=0 THEN fehler 0,30,50:RETURN
bo       GOSUB suchbegriff
U1       IF z$="" THEN RETURN
VN       su$=UCASE$(z$)
cm       loeschen
8L       fenster 0,30,350,75
rz       fenster 24,116,350,75
A7       information 380,30
eI       FOR i=1 TO mv
Hr6        TIMER OFF
Nm         COLOR 3,5
On         pixlocate 572,78:PRINT USING "####";i
ms         TIMER ON
97         LINE ((i-1)*216/mmv+393,90)-((i-1)*216/mmv+393,96),
           3
DX         LINE (i*216/mmv+393,90)-(i*216/mmv+393,96),4
nE         IF INSTR(UCASE$(v$(i,0)),su$)=0 AND INSTR(UCASE$(v$
           (i,1)),su$)=0 THEN nextvok
Tw         LINE (11,38)-(338,102),2,bf
Nm         LINE (35,124)-(360,189),2,bf
GY         z$=v$(i,0)
io         vokausgabe 11,46,z$
Ng         z$=v$(i,1)
Ib         vokausgabe 35,132,z$
Om         frage 390,115,51
Tp         IF aw THEN vokgef
wI         LINE (390,110)-(630,170),1,bf
qL2    nextvok:
SX4      NEXT
ah       fehler 390,115,52
1N       RETURN
mt2    vokgef:
204      LINE (390,110)-(630,170),1,bf
hp       fehler 390,115,53
pR       RETURN
vP2    suchbegriff:
g54      fenster 0,30,632,47
hH       TIMER OFF
Yu       COLOR 3,2
qM       pixlocate 12,46:PRINT "Bitte einen Suchbegriff eingeb
         en:"
CI       TIMER ON
pu       LINE (12,56)-(611,66),5,bf
Y8       rahmen 12,56,600,11
wp       z$=""
bf       stringeingabe 16,64,73,5
Oc       RETURN
NBO  REM Vokabel auf Diskette suchen
Y42    disksuche:
IS4      loeschen
vm       IF md=0 THEN fehler 0,30,31:RETURN
Qm       fenster 0,30,632,190
vV       TIMER OFF
m8       COLOR 3,2
vF       pixlocate 12,47:PRINT "Bitte Dateien markieren, die d
         urchsucht werden sollen:"
CY       FOR i=1 TO md
iV6        dm(i)=0
uz4      NEXT
5e       y=54
Gc       FOR i=1 TO md
wc6        y=y+10
Tw         pixlocate 20,y:PRINT d$(i)
pj4      IF y<176 THEN NEXT
Ln       kasten 16,198,121,15
U7       kasten 147,198,121,15
DN       kasten 280,198,121,15
NR       kasten 432,198,180,15
Fe       COLOR 3,5
PA       pixlocate 28,208:PRINT "letzte Datei"
ry       pixlocate 156,208:PRINT "nächste Datei"
1k       pixlocate 284,208:PRINT "Suche beginnen"
8G       pixlocate 442,208:PRINT "Zurück zum Hauptmenü"
5R       COLOR 3,2
io       TIMER ON
2V       n=1:e=1:be=0
ze       WHILE MOUSE(0)<0:WEND
Bd2    abfrage9:
714      WHILE MOUSE(0)=0:WEND
Di       x=MOUSE(3):y=MOUSE(4)
if       IF y<54 OR y>212 THEN abfrage9
5v       IF y<198 THEN aw=INT((y-47)/10):GOTO datmark
mz       IF x>431 THEN be=1:RETURN
fy       IF x>279 THEN voksuchen
w3       IF md<14 THEN abfrage9
vE       IF x>146 THEN n=n+1 ELSE IF x>15 THEN n=n-1
PY       IF n=0 THEN n=1:GOTO abfrage9 ELSE IF n+12>md THEN n
         =md-12:GOTO abfrage9
T3       TIMER OFF
Kg       COLOR 3,2
kV       IF n<e THEN datzurueck
1a       SCROLL (12,56)-(190,185),0,-10
Ii       IF dm(n+12) THEN COLOR 2,3 ELSE COLOR 3,2
BN       pixlocate 12,184:PRINT " " d$(n+12) SPACE$(21-LEN(d$(
         n+12)))
17       TIMER ON
4u       e=n
34       GOTO abfrage9
OK2    datzurueck:
VP4      SCROLL (12,56)-(190,185),0,10
3U       IF dm(n) THEN COLOR 2,3 ELSE COLOR 3,2
6g       pixlocate 12,64:PRINT " " d$(n) SPACE$(21-LEN(d$(n)))
8E       TIMER ON
B1       e=n
AB       GOTO abfrage9
1G2    datmark:
9A4      IF aw+n-1>md THEN abfrage9
hN       dm(aw+n-1)=1-dm(aw+n-1)
mM       TIMER OFF
r7       IF dm(aw+n-1) THEN COLOR 2,3 ELSE COLOR 3,2
aP       pixlocate 12,54+aw*10:PRINT " " d$(aw+n-1) SPACE$(21-
         LEN(d$(aw+n-1)))
f1       COLOR 3,2
IO       TIMER ON
YD       WHILE MOUSE(0)<0:WEND
KL       GOTO abfrage9
yd2    voksuchen:
KU4      loeschen
Na       GOSUB suchbegriff
Gn       IF z$="" THEN RETURN
H9       su$=UCASE$(z$)
OY       loeschen
u7       fenster 0,30,350,75
d1       fenster 24,116,350,75
wt       information 380,30
2c       TIMER OFF
8X       COLOR 3,5
97       pixlocate 572,64:PRINT "   0"
OM       pixlocate 572,78:PRINT "   0"
Ye       TIMER ON
cG       LINE (394,90)-(609,96),5,bf
Mi       FOR i=1 TO md
4L6        IF dm(i)=0 THEN nextdat
Rv         OPEN d$(i)+"."+LEFT$(na$(anw),3) FOR INPUT AS #1
qa         INPUT #1,e
ko         FOR j=1 TO e
Rd8          FOR k=1 TO 3
1pA            INPUT #1,e$
9E8          NEXT
AF6        NEXT
wg         INPUT #1,e
Is         TIMER OFF
On         COLOR 3,5
bK         pixlocate 572,64:PRINT USING "####";e
nt         TIMER ON
rV         LINE (394,90)-(609,96),5,bf
8Y         LINE (393,90)-(e*216/mmv+393,96),3,bf
w0         FOR j=1 TO e
Pz8          TIMER OFF
Vu           COLOR 3,5
Bz           pixlocate 572,78:PRINT USING "####";j
u0           TIMER ON
```

Listing 2. (Fortsetzung)

```
m6            s=(j-1)*216/mmv+393:t=j*216/mmv+393
pU            LINE (s,90)-(s,96),3
4h            LINE (t,90)-(t,96),4
fB            LINE INPUT #1,e$
2e            LINE INPUT #1,s$
p2            FOR k=0 TO 5
aQA             INPUT #1,s
UZ8           NEXT
ch            IF INSTR(UCASE$(e$),su$)=0 AND INSTR(UCASE$(s$),su$)=0 THEN nextvok2
hA            LINE (11,38)-(338,102),2,bf
b0            LINE (35,124)-(360,189),2,bf
OE            z$=e$
w2            vokausgabe 11,46,z$
Yc            z$=s$
Wp            vokausgabe 35,132,z$
c0            frage 390,115,51
oB            IF aw THEN CLOSE 1:GOTO vokgef
AW            LINE (390,110)-(630,170),1,bf
eR2     nextvok2:
g16         NEXT
QA          CLOSE 1
Mw2     nextdat:
jo4       NEXT
ry        fehler 390,115,52
2e        RETURN
BD0   REM  Vokabeln abfragen
jm2     abfrage:
2F4       IF mv=0 THEN CALL loeschen:fehler 0,30,54:RETURN
Op        menue 100,70,210,17,20
Qn        IF be THEN RETURN
jA        IF aw=1 THEN einpraegen
H7        mbn=mbn+1
je        IF mbn>10 THEN mbn=10:FOR i=0 TO 9:FOR j=0 TO 2:bn$(i,j)=bn$(i+1,j):NEXT j,i
Vx        bn$(mbn,0)=dat$
p1        bn$(mbn,1)=LEFT$(TIME$,5)
fR        ON aw-1 GOTO uebersetz,fremdwort,schrift
QK0   REM  Vokabeln einpraegen
202     einpraegen:
Vf4       loeschen
6d        funktion 17
2F        fenster 0,30,350,75
1t        fenster 24,116,350,75
41        information 380,30
PA        fenster 400,116,232,67
Ex        kasten 412,127,200,11
3k        kasten 412,143,200,11
M7        kasten 412,165,200,11
Eo        TIMER OFF
Kj        COLOR 3,5
Dz        pixlocate 420,135:PRINT "letzte Vokabel"
vF        pixlocate 420,151:PRINT "nächste Vokabel"
r1        pixlocate 420,173:PRINT "Zurück zum Hauptmenü"
9V        COLOR 3,2
ms        TIMER ON
Tr        z$=v$(1,0)
dj        vokausgabe 11,46,z$
az        z$=v$(1,1)
DW        vokausgabe 35,132,z$
mU        n=1:e=1
Uy2     abfrage10:
8n4       WHILE MOUSE(0)<0:WEND
Fq        WHILE MOUSE(0)=0:WEND
Lq        x=MOUSE(3):y=MOUSE(4)
Fy        IF x<413 OR x>611 OR y<125 OR y>177 THEN abfrage10
0G        IF y>164 THEN RETURN
n1        IF y>140 THEN n=n+1 ELSE n=n-1
gk        IF n<1 THEN n=mv ELSE IF n>mv THEN n=1
Y8        TIMER OFF
e3        COLOR 3,5
W0        pixlocate 572,78:PRINT USING "####";n
39        TIMER ON
aY        s=e*216/mmv+393:t=n*216/mmv+393
7x        e=n
ze        LINE (s,90)-(s,96),3
Er        LINE (t,90)-(t,96),4
1E        LINE (11,38)-(338,102),2,bf
f4        LINE (35,124)-(360,189),2,bf
nA        z$=v$(n,0)
06        vokausgabe 11,46,z$
uI        z$=v$(n,1)
at        vokausgabe 35,132,z$
09        GOTO abfrage10
Zy0   REM  Uebersetzung abfragen
gd2     uebersetz:
rP4       funktion 18
C3        ab=1
Jb        GOTO abfragen
NT2     fremdwort:
xW4       funktion 19
B1        ab=0
DW2     abfragen:
jL4       menue 140,93,214,21,23
Ro        IF be THEN RETURN
Je        re=aw
41        ausgmaske
x6        fenster 390,116,241,75
xf        kasten 402,126,210,11
cG        kasten 402,140,210,11
za        kasten 402,156,72,11
C0        kasten 540,156,72,11
Bw        kasten 402,174,210,11
6g        TIMER OFF
Cb        COLOR 3,5
nh        pixlocate 410,134:PRINT "insgesamt abgefragt:   0"
G0        pixlocate 410,148:PRINT "davon gewußt        :   0"
TT        pixlocate 410,164:PRINT "richtig"
OR        pixlocate 552,164:PRINT "falsch"
hq        pixlocate 410,182:PRINT "Zurück zum Hauptmenü"
3P        COLOR 3,2
gm        TIMER ON
ne        e=1
Lm2     bestfolge:
324       n=0
KV        ON re GOSUB reihenfolge,gemischt,koennen
NA2     naechstvok:
dh4       n=n+1
G9        IF n>mv THEN bestfolge
Ru        LINE (11,38)-(338,102),2,bf
Lk        LINE (35,124)-(360,189),2,bf
Fr        z$=v$(re(n),1-ab)
gm        vokausgabe 11,46,z$
32        IF ab THEN re1=0 ELSE re1=2
R1        TIMER OFF
Xw        COLOR 3,5
PH        pixlocate 572,78:PRINT USING "####";n
SQ        s=e*216/mmv+393:t=n*216/mmv+393
qV        LINE (s,90)-(s,96),3
51        LINE (t,90)-(t,96),4
1r        e=n
KT        pixlocate 570,134:PRINT USING "####";st(re(n),re1)
kG        pixlocate 570,148:PRINT USING "####";st(re(n),re1+1)
Qm        COLOR 3,2
39        TIMER ON
Jy        WHILE MOUSE(0)<0:WEND
Q1        WHILE MOUSE(0)=0:WEND
W1        x=MOUSE(3):y=MOUSE(4)
51        IF x>400 THEN IF y>170 THEN abfrbeenden
vZ        z$=v$(re(n),ab)
Wp        vokausgabe 35,132,z$
rM2     abfrage11:
Q54       WHILE MOUSE(0)<0:WEND
X8        WHILE MOUSE(0)=0:WEND
d8        x=MOUSE(3):y=MOUSE(4)
GA        IF x<402 OR x>611 OR y<154 OR y>190 THEN abfrage11
E1        IF y>170 THEN abfrbeenden
H6        IF x<506 THEN st(re(n),re1+1)=st(re(n),re1+1)+1
W2        st(re(n),re1)=st(re(n),re1)+1
k6        GOTO naechstvok
182     abfrbeenden:
ly4       bn$(mbn,2)=LEFT$(TIME$,5)
d9        IF lad THEN st=1:z$=d$(lad):GOTO datspeichern
5h        RETURN
cG0   REM  Schrift ueben
062     schrift:
pB4       funktion 20
1N        menue 140,93,214,21,23
Tq        IF be THEN RETURN
Lg        re=aw
```

```
734 6k      ausgmaske
735 yX      franzfont 402,116
736 kp      fenster 0,196,350,35
737 4e      TIMER OFF
738 vH      COLOR 3,2
739 HK      pixlocate 12,211:PRINT "insgesamt abgefragt:"
740 f6      pixlocate 12,225:PRINT "davon gewußt       :";
741 eH      kasten 180,203,50,11
742 31      kasten 180,217,50,11
743 Fe      COLOR 3,5
744 fJ      kasten 260,203,70,11
745 4n      kasten 260,217,70,11
746 Oz      pixlocate 268,211:PRINT "richtig"
747 Xy      pixlocate 268,225:PRINT "falsch";
748 hn      TIMER ON
749 wz      ab=2:e=1
750 Qn2   bestfolge2:
751 434     n=0
752 LW      ON re GOSUB reihenfolge,gemischt,koennen
753 OT2   vokeingeben:
754 ei4     n=n+1
755 Rc      IF n>mv THEN bestfolge2
756 Sv      LINE (11,38)-(338,102),2,bf
757 Ml      LINE (35,124)-(360,189),2,bf
758 5c      z$=v$(re(n),1)
759 hn      vokausgabe 11,46,z$
760 R1      TIMER OFF
761 Xw      COLOR 3,5
762 PH      pixlocate 572,78:PRINT USING "####";n
763 SQ      s=e*216/mmv+393:t=n*216/mmv+393
764 qV      LINE (s,90)-(s,96),3
765 5i      LINE (t,90)-(t,96),4
766 1r      e=n
767 Xx      pixlocate 188,211:PRINT USING "####";st(re(n),4)
768 Ox      pixlocate 188,225:PRINT USING "####";st(re(n),5);
769 Qm      COLOR 3,2
770 39      TIMER ON
771 qP      WHILE INKEY$<>"":WEND
772 mf      z$=""
773 km      stringeingabe 35,132,40,2
774 oY      z$=UCASE$(z$)
775 9G      IF z$="" THEN abfrbeenden
776 JD      st(re(n),4)=st(re(n),4)+1
777 8J      e$=UCASE$(v$(re(n),0))
778 bG      IF e$=z$ THEN richtig
779 uX2   teilwort:
780 oE4     s=INSTR(e$,",")
781 Bu      IF s=0 THEN s=LEN(e$)+1
782 kq      IF LEFT$(e$,s-1)=z$ THEN richtig
783 hS      e$=MID$(e$,s+1)
784 MW      IF e$<>"" THEN teilwort
785 mE      LINE (262,204)-(327,212),5,bf
786 Lv      LINE (262,218)-(327,226),3,bf
787 sS      TIMER OFF
788 yN      COLOR 3,5
789 hg      pixlocate 268,211:PRINT "richtig"
790 wL      COLOR 5,3
791 Fg      pixlocate 268,225:PRINT "falsch";
792 vK      LINE (35,124)-(360,189),2,bf
793 oA      COLOR 3,2
794 b7      z$=v$(re(n),0)
795 p8      vokausgabe 35,132,z$
796 TZ      TIMER ON
797 uN      FOR s=1 TO 3000:NEXT
798 kP      WHILE MOUSE(0)<0:WEND
799 7X      WHILE MOUSE(0)=0 AND INKEY$="":WEND
800 ZD      GOTO vokeingeben
801 pt2   richtig:
802 zP4     LINE (262,204)-(327,212),3,bf
803 gI      LINE (262,218)-(327,226),5,bf
804 9j      TIMER OFF
805 Ba      COLOR 5,3
806 yx      pixlocate 268,211:PRINT "richtig"
807 Hg      COLOR 3,5
808 Wx      pixlocate 268,225:PRINT "falsch";
809 4Q      COLOR 3,2
810 hn      TIMER ON
811 1s      st(re(n),5)=st(re(n),5)+1
812 1P      GOTO vokeingeben
813 860 REM  Statistik ausgeben
814 vi2   statistik:
815 Gy4     funktion 8
816 Vj      IF mv=0 THEN CALL loeschen:fehler 0,30,55:RETURN
817 Of      menue 100,90,250,24,26
818 rE      IF be THEN RETURN
819 68      funktion 23+aw
820 MX      n=aw
821 QG      IF n=1 THEN ab=0 ELSE IF n=2 THEN ab=2 ELSE ab=4
822 dX      fenster 0,30,632,197
823 iJ      y=38
824 T3      TIMER OFF
825 sW      FOR i=1 TO mv
826 SA6       y=y+12
827 Mi        COLOR 3,2
828 8o        pixlocate 12,y:PRINT USING "###.";i
829 Rk        pixlocate 56,y:PRINT LEFT$(v$(i,0),31)
830 tp        kasten 320,y-8,98,11
831 f4        COLOR 3,5
832 W4        pixlocate 324,y:PRINT USING "###";st(i,ab+1);
833 M5        PRINT " von " USING "###";st(i,ab)
834 FL        kasten 430,y-8,180,11
835 qO        IF st(i,ab) THEN LINE (431,y-6)-(431+st(i,ab+1)*176
          /st(i,ab),y),3,bf
836 4t4     IF y<180 THEN NEXT
837 vj      e=0:e1=0
838 5j      FOR i=1 TO mv
839 AY6       e=e+st(i,ab)
840 nU        e1=e1+st(i,ab+1)
841 ej4     NEXT
842 dm      IF e THEN s=e1*100/e ELSE s=0
843 Aj      kasten 12,192,280,11
844 Ds      kasten 304,192,114,11
845 EG      kasten 430,192,180,11
846 f8      kasten 12,210,134,11
847 Oj      kasten 158,210,134,11
848 eX      kasten 430,210,180,11
849 xM      COLOR 3,5
850 Vw      pixlocate 18,200:PRINT "Durchschnittlich " USING"##
        # % gewußt";s
851 QH      pixlocate 308,200:PRINT USING "####";e1;
852 3I      PRINT " von " USING "####";e
853 71      IF e THEN LINE (431,194)-(431+e1*176/e,200),3,bf
854 jB      pixlocate 22,218:PRINT "letzte Vokabel"
855 Hf      pixlocate 164,218:PRINT "nächste Vokabel"
856 nu      pixlocate 440,218:PRINT "Zurück zum Hauptmenü"
857 N5      n=1:e=1
858 iN      WHILE MOUSE(0)<0:WEND
859 Gm2   abfrage12:
860 qR4     WHILE MOUSE(0)=0:WEND
861 wR      x=MOUSE(3):y=MOUSE(4)
862 cN      IF x<12 OR x>609 OR y<210 OR y>220 THEN abfrage12
863 k3      IF x>429 THEN RETURN
864 5B      IF mv<13 THEN abfrage12
865 ZP      IF x>292 THEN abfrage12
866 Sh      IF x>151 THEN n=n+1 ELSE n=n-1
867 es      IF n=0 THEN n=1:GOTO abfrage12 ELSE IF n+11>mv THEN
        n=mv-11:GOTO abfrage12
868 B1      TIMER OFF
869 20      COLOR 3,2
870 hk      IF n<e THEN nextstat
871 gM      SCROLL (12,42)-(609,184),0,-12
872 LQ      pixlocate 12,182:PRINT USING "###.";n+11
873 KY      pixlocate 56,182:PRINT LEFT$(v$(n+11,0),31)
874 RJ      kasten 320,174,98,11
875 Nm      COLOR 3,5
876 91      pixlocate 324,182:PRINT USING "###";st(n+11,ab+1);
877 81      PRINT " von " USING "###";st(n+11,ab)
878 np      kasten 430,174,180,11
879 vc      IF st(n+11,ab) THEN LINE (431,176)-(431+st(n+11,ab+1)
        *176/st(n+11,ab),182),3,bf
880 pv      TIMER ON
881 si      e=n
882 ny      GOTO abfrage12
883 xv2   nextstat:
884 pp4     SCROLL (12,38)-(609,184),0,12
885 9T      pixlocate 12,50:PRINT USING "###.";n
886 2Z      pixlocate 56,50:PRINT LEFT$(v$(n,0),31)
887 9z      kasten 320,42,98,11
888 az      COLOR 3,5
889 LX      pixlocate 324,50:PRINT USING "###";st(n,ab+1);
890 1A      PRINT " von " USING "###";st(n,ab)
```

Listing 2. (Fortsetzung)

```
      kasten 430,42,180,11
      IF st(n,ab) THEN LINE (431,44)-(431+st(n,ab+1)*176/st
      (n,ab),50),3,bf
      TIMER ON
      e=n
      GOTO abfrage12
REM  Programm beenden
  ende:
      funktion 9
      loeschen
      IF va THEN frage 0,30,56:IF aw=0 THEN RETURN ELSE anw
      speichern
      frage 0,30,57
      IF aw=0 THEN RETURN
  anwspeichern:
      IF anw=0 THEN beenden
      abn$(mabn,2)=LEFT$(TIME$,5)
      OPEN "BENUTZT."+LEFT$(na$(anw),3) FOR OUTPUT AS 1
      PRINT #1,mabn
      FOR i=1 TO mabn
        FOR j=0 TO 2
          PRINT #1,abn$(i,j)
        NEXT
      NEXT
      CLOSE 1
      IF fae THEN GOSUB faspeichern
  beenden:
      MENU RESET
      COLOR 3,1
      CLS
      END
REM  Sonderfunktionen
  sonderfunktion:
      funktion 10
      menue 90,80,230,59,61
      IF be THEN RETURN
      ON aw GOTO farbwechsel,datum,benutzung
REM  Farben aendern
  farbwechsel:
      loeschen
      funktion 59
      fenster 0,30,348,112
      rahmen 14,44,248,19
      FOR i=1 TO 7
        LINE (i*34-12,48)-(i*34+16,58),i,bf
      NEXT
      rahmen 20,47,33,13
      FOR i=1 TO 3
        kasten 14,i*16+56,248,11
        kasten 276,i*16+56,50,11
        FOR j=1 TO 16
          e=248/16*j+5
          LINE (e,i*16+65)-(e+1,i*16+66),3,b
        NEXT
      NEXT
      kasten 150,124,176,11
      TIMER OFF
      COLOR 3,5
      pixlocate 284,80:PRINT "rot"
      pixlocate 284,96:PRINT "grün"
      pixlocate 284,112:PRINT "blau"
      pixlocate 158,132:PRINT "Zurück zum Hauptmenü"
      TIMER ON
      FOR i=1 TO 3
        e=farb(1,i-1)*15.5
        LINE (e+4,i*16+58)-(e+7,i*16+63),4,bf
      NEXT
      n=1
  abfrage13:
      WHILE MOUSE(0)<0:WEND
      WHILE MOUSE(0)=0:WEND
      x=MOUSE(3):y=MOUSE(4)
      IF x<14 OR x>325 OR y<44 OR y>141 THEN abfrage13
      IF y>123 THEN IF x>148 THEN farbbeenden
      IF x>260 THEN abfrage13
      IF y<63 THEN farbaendern
      x=INT((x+2)/15.5)
      IF y>101 THEN y=2 ELSE IF y>85 THEN y=1 ELSE y=0
      e=farb(n,y)*15.5+4
      LINE (e,y*16+74)-(e+3,y*16+79),5,bf
      farb(n,y)=x
      IF n=1 THEN farb(0,y)=x
```

```
      fae=1
      e=farb(n,y)*15.5+4
      LINE (e,y*16+74)-(e+3,y*16+79),4,bf
      PALETTE n,f!(farb(n,0)),f!(farb(n,1)),f!(farb(n,2))
      IF n=1 THEN PALETTE 0,f!(farb(n,0)),f!(farb(n,1)),f!(
      farb(n,2))
      GOTO abfrage13
  farbaendern:
      x=INT((x+14)/34)
      IF x<1 THEN x=1 ELSE IF x>7 THEN x=7
      LINE (n*34-14,47)-(n*34+18,59),2,b
      LINE (n*34-13,47)-(n*34-13,59),2
      LINE (n*34+17,47)-(n*34+17,59),2
      FOR i=0 TO 2
        e=farb(n,i)*15.5+4
        LINE (e,i*16+74)-(e+3,i*16+79),5,bf
      NEXT
      n=x
      rahmen n*34-14,47,33,13
      FOR i=0 TO 2
        e=farb(n,i)*15.5
        LINE (e+4,i*16+74)-(e+7,i*16+79),4,bf
      NEXT
      GOTO abfrage13
  farbbeenden:
      FOR i=0 TO 7
        FOR j=0 TO 2
          fa(anw,i,j)=farb(i,j)
        NEXT
      NEXT
      RETURN
REM  Abfragedaten ausgeben
  datum:
      loeschen
      funktion 60
      IF mv=0 THEN fehler 0,30,62:RETURN
      IF lad=0 THEN fehler 0,30,63:RETURN
      IF mbn=0 THEN fehler 0,30,64:RETURN
      fenster 0,30,450,mbn*12+46
      TIMER OFF
      COLOR 3,2
      pixlocate 12,47:PRINT "Abfragedaten zur Vokabeldatei:
      "
      kasten 260,39,174,11
      COLOR 3,5
      pixlocate 268,47:PRINT d$(lad)
      y=52
      COLOR 3,2
      FOR i=1 TO mbn
        y=y+12
        pixlocate 12,y:PRINT bn$(i,0)
        pixlocate 110,y:PRINT "von  " bn$(i,1) " Uhr  bis
        " bn$(i,2) " Uhr"
      NEXT
      kasten 389,mbn*12+60,44,11
      COLOR 3,5
      pixlocate 402,mbn*12+68:PRINT "OK"
      COLOR 3,2
      TIMER ON
  abfrage14:
      WHILE MOUSE(0)<0:WEND
      WHILE MOUSE(0)=0:WEND
      x=MOUSE(3):y=MOUSE(4)
      IF x<389 OR x>432 THEN abfrage14
      IF y-mbn*12<60 OR y-mbn*12>70 THEN abfrage14
      RETURN
REM  Benutzungsdaten ausgeben
  benutzung:
      loeschen
      funktion 61
      IF man=0 THEN fehler 0,30,65:RETURN
      fenster 0,30,372,mabn*12+46
      TIMER OFF
      COLOR 3,2
      pixlocate 12,47:PRINT "Benutzungsdaten von"
      kasten 180,39,174,11
      COLOR 3,5
      pixlocate 186,47:PRINT na$(anw)
      y=52
      COLOR 3,2
      FOR i=1 TO mabn
        y=y+12
```

```
1050 qt         pixlocate 12,y:PRINT abn$(i,0)
1051 tJ         pixlocate 110,y:PRINT "von  " abn$(i,1) " Uhr  bis
                 ";
1052 cl         IF abn$(i,2)<>"" THEN PRINT abn$(i,2) " Uhr" ELSE
                 PRINT "jetzt"
1053 494      NEXT
1054 4F       kasten 310,mabn*12+60,44,11
1055 Hg       COLOR 3,5
1056 vc       pixlocate 323,mabn*12+68:PRINT "OK"
1057 4Q       COLOR 3,2
1058 hn       TIMER ON
1059 jI2    abfrage15:
1060 yd4      WHILE MOUSE(0)<0:WEND
1061 5g       WHILE MOUSE(0)=0:WEND
1062 Bg       x=MOUSE(3):y=MOUSE(4)
1063 GO       IF x<309 OR x>352 THEN abfrage15
1064 RG       IF y-mabn*12<60 OR y-mabn*12>70 THEN abfrage15
1065 X9       RETURN
1066 ZNO  REM  Unterprogramme
1067 FX   REM  Menuefunktion ausgeben
1068 OZ2    SUB funktion (n) STATIC
1069 Q04      TIMER OFF
1070 Hd       COLOR 3,2
1071 3d       pixlocate 10,16:PRINT t$(n) SPACE$(40-LEN(t$(n)))
1072 v1       TIMER ON
1073 LN       END SUB
1074 rKO  REM  Bildschirm loeschen
1075 up2    SUB loeschen STATIC
1076 qX4      LINE (0,23)-(632,230),1,bf
1077 PR       END SUB
1078 aFO  REM  Rahmen ausgeben
1079 cF2    SUB rahmen (x,y,l,b) STATIC
1080 IW4      LINE (x,y)-(x+l-1,y+b-1),3,b
1081 TG       LINE (x+1,y)-(x+1,y+b-2),3
1082 mO       LINE (x+l-2,y)-(x+l-2,y+b-2),3
1083 VX       END SUB
1084 xSO  REM  Fenster ausgeben
1085 VS2    SUB fenster (x,y,l,b) STATIC
1086 ST4      LINE (x,y+4)-(x+l-8,y+b-1),2,bf
1087 HB       LINE (x,y+4)-(x+l-8,y+b-1),3,b
1088 iV       LINE (x+1,y+5)-(x+1,y+b-2),3
1089 TI       LINE (x+l-9,y+5)-(x+l-9,y+b-2),3
1090 ay       LINE (x+8,y)-(x+l-1,y+3),3,bf
1091 wS       LINE (x+l-7,y+4)-(x+l-1,y+b-5),3,bf
1092 eg       END SUB
1093 s10  REM  Kasten ausgeben
1094 AX2    SUB kasten (x,y,l,b) STATIC
1095 Fd4      LINE (x,y)-(x+l-1,y+b-1),5,bf
1096 w4       rahmen x,y,l,b
1097 jl       END SUB
1098 UEO  REM  Fenster fuer Vokabelausgabe
1099 e62    SUB vokfenster STATIC
1100 cy4      fenster 0,30,632,80
1101 wW       TIMER OFF
1102 47       pixlocate 12,46;PRINT "Fremdwort:"
1103 UH       pixlocate 12,84:PRINT "Übersetzung:"
1104 RX       TIMER ON
1105 gO       kasten 12,53,600,11
1106 jT       kasten 12,91,600,11
1107 tv       END SUB
1108 NSO  REM  Ausgabemaske erstellen
1109 RM2    SUB ausgmaske STATIC
1110 Vf4      loeschen
1111 1E       fenster 0,30,350,75
1112 ks       fenster 24,116,350,75
1113 30       information 380,30
1114 02       END SUB
1115 EwO  REM  Informationsfenster ausgeben
1116 rx2    SUB information (x,y) STATIC
1117 K34      SHARED mv,mmv
1118 qL       fenster x,y,252,75
1119 Eo       TIMER OFF
1120 5R       COLOR 3,2
1121 IY       pixlocate x+12,y+18:PRINT "Information:
1122 OG       pixlocate x+12,y+34:PRINT "Vokabelanzahl gesamt:"
1123 x7       pixlocate x+12,y+48:PRINT "Vokabelnummer        :"
1124 S1       kasten x+188,y+26,44,11
1125 Fk       kasten x+188,y+40,44,11
1126 Qp       COLOR 3,5
1127 8W       pixlocate x+192,y+34:PRINT USING "####";mv
1128 Wh       pixlocate x+192,y+48:PRINT "   1"
1129 qw       TIMER ON
1130 yq       kasten x+12,y+58,220,11
1131 kO       LINE (x+13,y+60)-(x+13+mv*216/mmv,y+66),3,bf
1132 GH       LINE (x+14,y+60)-(x+14,y+66),4
1133 JL       END SUB
1134 fLO  REM  Cursor auf Grafikkoordinaten positionieren
1135 gf2    SUB pixlocate (x,y) STATIC
1136 3E4      s&=WINDOW(8)+36:t&=WINDOW(8)+38
1137 5Z       POKEW s&,x:POKEW t&,y
1138 OQ       END SUB
1139 pEO  REM  Menuefunktion
1140 Di2    SUB menue (x,y,l,at,et) STATIC
1141 ja4      SHARED aw,be
1142 2b       b=(et-at+1)*14+12:be=0
1143 3L       fenster x,y,l,b
1144 dD       TIMER OFF
1145 Uq       COLOR 3,2
1146 Y1       FOR i=at TO et
1147 cA6        pixlocate x+12,(i-at)*14+y+18:PRINT t$(i)
1148 bg4      NEXT
1149 AG       TIMER ON
1150 5P2    abfrage1:
1151 R64      WHILE MOUSE(0)<0:WEND
1152 Y9       WHILE MOUSE(0)=0:WEND
1153 bR       xp=MOUSE(3):yp=MOUSE(4)
1154 NW       IF xp<x+1 OR xp>x+l-8 OR yp<y+4 OR yp>y+b-1 THEN
              be=1:EXIT SUB
1155 Jq       aw=INT((yp-y+4)/14)
1156 74       IF aw<1 OR aw>et-at+1 THEN abfrage1
1157 qQ       TIMER OFF
1158 k7       COLOR 4,2
1159 hH       pixlocate x+12,aw*14+y+4:PRINT t$(at+aw-1)
1160 LR       TIMER ON
1161 ln       END SUB
1162 cKO  REM  Fragestellung
1163 er2    SUB frage (x,y,n) STATIC
1164 Tv4      SHARED aw
1165 M2       e=1:e1=1:s=1:l=0
1166 jd       WHILE e1
1167 PM6        e1=INSTR(e,t$(n),"*")
1168 uP         IF e1-e>l THEN l=e1-e
1169 qy         IF e1 THEN e=e1+1:s=s+1
1170 J74      WEND
1171 Qw       IF LEN(t$(n))-e+1>l THEN l=LEN(t$(n))-e+1
1172 EX       l=l*8+32
1173 11       fenster x,y,l,s*12+24
1174 9F       e$=t$(n):e=y+4:e1=1
1175 81       TIMER OFF
1176 tn       WHILE e1
1177 Z16        e1=INSTR(e$,"*"):e=e+12
1178 R8         IF e1 THEN pixlocate x+12,e:PRINT LEFT$(e$,e1-1)
1179 NX         e$=MID$(e$,e1+1)
1180 TH4      WEND
1181 1F       s=s+1
1182 XR       pixlocate x+12,e:PRINT e$
1183 XE       FOR i=x+l-108 TO x+l-58 STEP 50
1184 LZ6        LINE (i,y+s*12-2)-(i+36,y+s*12+7),5,bf
1185 es         LINE (i,y+s*12-2)-(i+36,y+s*12+7),3,b
1186 g4         LINE (i-1,y+s*12-2)-(i-1,y+s*12+7),3
1187 9V         LINE (i+37,y+s*12-2)-(i+37,y+s*12+7),3
1188 FK4      NEXT
1189 PU       e=e+13
1190 Sr       COLOR 3,5
1191 7W       pixlocate x+l-98,e:PRINT "ja"
1192 RK       pixlocate x+l-55,e:PRINT "nein"
1193 sy       TIMER ON
1194 vH2    abfrage3:
1195 9o4      WHILE MOUSE(0)<0:WEND
1196 Gr       WHILE MOUSE(0)=0:WEND
1197 J9       xp=MOUSE(3):yp=MOUSE(4)
1198 CV       IF xp<x+l-109 OR xp>x+l-22 OR yp<e-7 OR yp>e+2 TH
              EN abfrage3
1199 Bv       IF xp<x+l-63 THEN aw=1 ELSE aw=0
1200 OQ       END SUB
1201 KPO  REM  Fehlermeldung
1202 KH2    SUB fehler (x,y,n) STATIC
1203 ye4      e=1:e1=1:s=1:l=0
1204 LF       WHILE e1
1205 1y6        e1=INSTR(e,t$(n),"*")
1206 W1         IF e1-e>l THEN l=e1-e
1207 Sa         IF e1 THEN e=e1+1:s=s+1
```

Listing 2. (Fortsetzung)

```
1208 vj4      WEND
1209 2Y       IF LEN(t$(n))-e+1>l THEN l=LEN(t$(n))-e+1
1210 q9       l=l*8+32
1211 Pg       fenster x,y,l,s*12+25
1212 lr       e$=t$(n):e=y+4:e1=1
1213 kK       TIMER OFF
1214 bx       COLOR 3,2
1215 WQ       WHILE e1
1216 Ce6        e1=INSTR(e$,"*"):e=e+12
1217 41         IF e1 THEN pixlocate x+12,e:PRINT LEFT$(e$,e1-1)
1218 0A         e$=MID$(e$,e1+1)
1219 6u4      WEND
1220 es       s=s+1
1221 A4       pixlocate x+12,e:PRINT e$
1222 AY       LINE (x+l-54,y+s*12-2)-(x+l-18,y+s*12+8),5,bf
1223 JB       LINE (x+l-54,y+s*12-2)-(x+l-18,y+s*12+8),3,b
1224 JC       LINE (x+l-55,y+s*12-2)-(x+l-55,y+s*12+8),3
1225 Ap       LINE (x+l-17,y+s*12-2)-(x+l-17,y+s*12+8),3
1226 2R       COLOR 3,5
1227 39       e=e+14
1228 NF       pixlocate x+l-44,e:PRINT "OK"
1229 SY       TIMER ON
1230 Zw2    abfrage4:
1231 j04      WHILE MOUSE(0)<0:WEND
1232 qR       WHILE MOUSE(0)=0:WEND
1233 tj       xp=MOUSE(3):yp=MOUSE(4)
1234 WN       IF xp<x+l-55 OR xp>x+l-17 OR yp<e-8 OR yp>e+2 THE
              N abfrage4
1235 xz       END SUB
1236 k10  REM  Datei auswaehlen
1237 M22    SUB dateiauswahl STATIC
1238 yL4      SHARED md,be,aw
1239 g2       fenster 0,30,632,190
1240 B1       TIMER OFF
1241 20       COLOR 3,2
1242 GL       pixlocate 16,47:PRINT "Vokabeldatei"
1243 lv       pixlocate 250,47:PRINT "Erstellungsdatum"
1244 Js       y=54
1245 Uq       FOR i=1 TO md
1246 Aq6        y=y+10
1247 xV         pixlocate 16,y:PRINT d$(i)
1248 PE         pixlocate 250,y:PRINT da$(i)
1249 4y4      IF y<176 THEN NEXT
1250 a2       kasten 16,198,121,15
1251 JM       kasten 147,198,121,15
1252 bf       kasten 432,198,180,15
1253 Ts       COLOR 3,5
1254 d0       pixlocate 28,208:PRINT "letzte Datei"
1255 5C       pixlocate 156,208:PRINT "nächste Datei"
1256 LT       pixlocate 442,208:PRINT "Zurück zum Hauptmenü"
1257 u0       TIMER ON
1258 Eh       n=1:e=1:be=0
1259 Bq       WHILE MOUSE(0)<0:WEND
1260 Ba2    abfrage6:
1261 Ju4      WHILE MOUSE(0)=0:WEND
1262 Pu       x=MOUSE(3):y=MOUSE(4)
1263 cl       IF y<54 OR y>212 THEN abfrage6
1264 2t       IF y<198 THEN aw=INT((y-47)/10):GOTO datwahl
1265 Xy       IF x>431 THEN be=1:EXIT SUB
1266 y2       IF md<14 THEN abfrage6
1267 6P       IF x>146 THEN n=n+1 ELSE IF x>15 THEN n=n-1
1268 CF       IF n=0 THEN n=1:GOTO abfrage6 ELSE IF n+12>md THEN n
              =md-12:GOTO abfrage6
1269 eE       TIMER OFF
1270 xJ       IF n<e THEN SCROLL (16,55)-(340,185),0,10:pixlocate
              16,64:PRINT d$(n):pixlocate 250,64:PRINT da$(n)
1271 3K       IF n>e THEN SCROLL (16,55)-(340,185),0,-10:pixlocate
               16,184:PRINT d$(n+12):pixlocate 250,184:PRINT da$(n+1
              2)
1272 9F       TIMER ON
1273 C2       e=n
1274 zx       GOTO abfrage6
1275 Jp2    datwahl:
1276 424      IF aw+n-1>md THEN abfrage6
1277 mM       TIMER OFF
1278 f1       COLOR 2,3
1279 TK       pixlocate 8,54+aw*10:PRINT " " d$(aw+n-1) SPACE$(29-L
              EN(d$(aw+n-1)))
1280 CI       pixlocate 242,54+aw*10:PRINT " " da$(aw+n-1) " "
1281 g2       COLOR 3,2
1282 JP       TIMER ON
1283 s9       aw=aw+n-1
1284 km       END SUB
1285 su0  REM  Vokabel auswaehlen
1286 232    SUB vokauswahl STATIC
1287 084      SHARED aw,be,mv
1288 Tp       fenster 0,30,632,190
1289 yY       TIMER OFF
1290 gJ       pixlocate 16,47:PRINT "Fremdwort"
1291 4B       pixlocate 320,47:PRINT "Übersetzung"
1292 5e       y=54
1293 sE       COLOR 3,2
1294 R5       FOR i=1 TO mv
1295 xd6        y=y+10
1296 8T         pixlocate 16,y:PRINT LEFT$(v$(i,0),37)
1297 Ap         pixlocate 320,y:PRINT LEFT$(v$(i,1),36)
1298 r14      IF y<176 THEN NEXT
1299 hG       kasten 16,198,137,15
1300 j1       kasten 163,198,137,15
1301 OS       kasten 432,198,180,15
1302 Gf       COLOR 3,5
1303 Am       pixlocate 28,208:PRINT "letzte Vokabel"
1304 P1       pixlocate 172,208:PRINT "nächste Vokabel"
1305 8G       pixlocate 442,208:PRINT "Zurück zum Hauptmenü"
1306 hn       TIMER ON
1307 1U       n=1:e=1:be=0
1308 yd       WHILE MOUSE(0)<0:WEND
1309 6X2    abfrage8:
1310 6h4      WHILE MOUSE(0)=0:WEND
1311 Ch       x=MOUSE(3):y=MOUSE(4)
1312 bc       IF y<54 OR y>212 THEN abfrage8
1313 FC       IF y<198 THEN aw=INT((y-47)/10):GOTO vokwahl
1314 K1       IF x>431 THEN be=1:EXIT SUB
1315 bX       IF mv<14 THEN abfrage8
1316 j0       IF x>162 THEN n=n+1 ELSE IF x>15 THEN n=n-1
1317 jW       IF n=0 THEN n=1:GOTO abfrage8 ELSE IF n+12>mv THEN n
              =mv-12:GOTO abfrage8
1318 R1       TIMER OFF
1319 Ie       COLOR 3,2
1320 db       IF n<e THEN SCROLL (16,56)-(611,185),0,10:pixlocate
              16,64:PRINT LEFT$(v$(n,0),37):pixlocate 320,64:PRINT L
              EFT$(v$(n,1),36)
1321 mQ       IF n>e THEN SCROLL (16,56)-(611,185),0,-10:pixlocate
               16,184:PRINT LEFT$(v$(n+12,0),37):pixlocate 320,184:P
              RINT LEFT$(v$(n+12,1),36)
1322 x3       TIMER ON
1323 0q       e=n
1324 vv       GOTO abfrage8
1325 ng2    vokwahl:
1326 o64      IF aw+n-1>mv THEN abfrage8
1327 aA       TIMER OFF
1328 Tp       COLOR 2,3
1329 Gv       pixlocate 8,54+aw*10:PRINT " " LEFT$(v$(aw+n-1,0),37)
               SPACE$(38-LEN(LEFT$(v$(aw+n-1,0),37)))
1330 J1       pixlocate 312,54+aw*10:PRINT " " LEFT$(v$(aw+n-1,1),3
              6) SPACE$(37-LEN(LEFT$(v$(aw+n-1,1),36)))
1331 Uq       COLOR 3,2
1332 7D       TIMER ON
1333 gx       aw=aw+n-1
1334 Ya       END SUB
1335 Ga0  REM  Zeichenkette eingeben
1336 yY2    SUB stringeingabe (x,y,mz,f) STATIC
1337 Bh4      SHARED z$
1338 1L       TIMER OFF
1339 Y0       IF z$<>"" THEN COLOR 3,f:pixlocate x,y:PRINT z$;:CO
              LOR f,4:PRINT " " ELSE COLOR f,4:pixlocate x,y:PRINT "
              "
1340 dz       COLOR 3,2
1341 GM       TIMER ON
1342 Fa2    abfrage2:
1343 4U4      e$=INKEY$:IF e$="" THEN abfrage2
1344 em       ac=ASC(e$)
1345 XK       IF ac=8 THEN delet
1346 zG       IF ac=13 THEN enter
1347 k8       IF LEN(z$)=mz THEN abfrage2
1348 Ia       IF ac>128 AND ac<139 THEN e$=CHR$(ff(ac-128))
1349 wW       TIMER OFF
1350 zB       COLOR 3,f
1351 cJ       pixlocate x+LEN(z$)*8,y:PRINT e$;
1352 Sf       COLOR f,4
1353 R8       PRINT " "
1354 3F       COLOR 3,f
1355 Ua       TIMER ON
1356 Ry       z$=z$+e$
```

```
    GOTO abfrage2
  delet:
    IF z$="" THEN abfrage2
    TIMER OFF
    COLOR f,4
    pixlocate x+LEN(z$)*8-8,y:PRINT " ";
    COLOR 3,f
    PRINT " "
    TIMER ON
    z$=LEFT$(z$,LEN(z$)-1)
    GOTO abfrage2
  enter:
    TIMER OFF
    COLOR 3,f
    pixlocate x+LEN(z$)*8,y:PRINT " "
    TIMER ON
    END SUB
REM  Vokabel aufgeteilt ausgeben
  SUB vokausgabe (x,y,e$) STATIC
    yp=y-10
    TIMER OFF
    COLOR 3,2
  kommasu:
    e=INSTR(e$,",")
    IF e=0 THEN yp=yp+10:IF yp<y+58 THEN pixlocate x,yp:
    PRINT LEFT$(e$,40):GOTO beendenausg ELSE beendenausg
    yp=yp+10
    IF yp>y+58 THEN beendenausg
    pixlocate x,yp:PRINT LEFT$(LEFT$(e$,e-1),40)
    e$=MID$(e$,e+1)
    GOTO kommasu
  beendenausg:
    TIMER ON
    END SUB
REM  franzoesische Sonderzeichen
  SUB franzfont (x,y) STATIC
    fenster x,y,230,83
    TIMER OFF
    COLOR 3,2
    pixlocate x+12,y+16:PRINT "franz. Zeichensatz:"
    xp=0:yp=1
    COLOR 3,5
    FOR i=1 TO 10
      xp=xp+1:IF xp=4 THEN xp=1:yp=yp+1
      s=x+xp*70-58:t=y+yp*14+10
      LINE (s,t)-(s+58,t+9),5,bf
      rahmen s,t,60,11
      pixlocate s+6,t+8:PRINT USING "F##: ";i;:PRINT CH
      R$(ff(i))
    NEXT
    TIMER ON
    END SUB
REM  Abfrage nach Reihenfolge festlegen
  reihenfolge:
    FOR s=1 TO mv
      re(s)=s
    NEXT
    RETURN
REM  Abfrage gemischt festlegen
  gemischt:
    FOR s=0 TO mv
      re(s)=0
    NEXT
    FOR s=1 TO INT(mv/2)
  zufall1:
      t=INT(RND(1)*mv)+1
      IF re(t) THEN zufall1
      re(t)=s
    NEXT
    t=INT(mv/2)
    FOR s=1 TO mv
      IF re(s)=0 THEN t=t+1:re(s)=t
    NEXT
    RETURN
REM  Abfrage nach Koennen festlegen
  koennen:
    IF ab=1 THEN u=0 ELSE IF ab=0 THEN u=2 ELSE u=4
    i=0:e1=0
    FOR s=1 TO mv
      i=i+st(s,u)
      e1=e1+st(s,u+1)
      vo(s)=0
    NEXT
    IF i+e1=0 THEN gemischt
    i=e1*100/i
    e1=0
    FOR s=1 TO mv
      IF INT(st(s,u+1)*100/st(s,u))<=i THEN e1=e1+1:re1(
      e1)=s:vo(s)=1
    NEXT
  auffuellen:
    i=i+10
    FOR s=1 TO mv
      IF INT(st(s,u+1)*100/st(s,u))>=i THEN zugut
      IF re1(e1)=s THEN zugut
      IF vo(s)>1 THEN zugut
      e1=e1+1:re1(e1)=s:vo(s)=vo(s)+1
      IF e1>=mv THEN aufbeenden
  zugut:
    NEXT
    GOTO auffuellen
  aufbeenden:
    FOR s=1 TO mv
      re(s)=0
    NEXT
    FOR s=1 TO mv/2
  zufall2:
      u=INT(RND(1)*mv)+1
      IF re(u) THEN zufall2
      IF re(u-1)=re1(s) OR re(u+1)=re1(s) THEN zufall2
      re(u)=re1(s)
    NEXT
    u=INT(mv/2)
    FOR s=1 TO mv
      IF re(s)=0 THEN u=u+1:re(s)=re1(u)
    NEXT
    RETURN
REM  Dateinamen laden
  SUB datnamenlad STATIC
    SHARED anw,be,md
    be=0
    e$="DATEIEN."+LEFT$(na$(anw),3)
    OPEN e$ FOR APPEND AS #1
    s=LOF(1)
    CLOSE 1
    IF s=0 THEN KILL e$:be=1:EXIT SUB
    OPEN "DATEIEN."+LEFT$(na$(anw),3) FOR INPUT AS 1
    INPUT #1,md
    FOR i=1 TO md
      LINE INPUT #1,d$(i)
      INPUT #1,da$(i)
    NEXT
    CLOSE 1
    END SUB
REM  Dateinamen speichern
  SUB datnamensp STATIC
    SHARED anw,md
    OPEN "DATEIEN."+LEFT$(na$(anw),3) FOR OUTPUT AS #1
    PRINT #1,md
    FOR i=1 TO md
      PRINT #1,d$(i)
      PRINT #1,da$(i)
    NEXT
    CLOSE 1
    END SUB
REM  Anwendernamen laden
  SUB anwenderladen STATIC
    SHARED man
    CHDIR "DF0:"
    OPEN "ANWENDER" FOR APPEND AS 1
    s=LOF(1)
    CLOSE 1
    IF s=0 THEN anwenderinit
    OPEN "ANWENDER" FOR INPUT AS 1
    INPUT #1,man
    FOR i=1 TO man
      LINE INPUT #1,na$(i)
      FOR j=0 TO 7
        FOR k=0 TO 2
          INPUT #1,fa(i,j,k)
        NEXT
```

Listing 2. (Fortsetzung)

```
1515 Wb6        NEXT
1516 Xc4      NEXT
1517 H1       CLOSE 1
1518 dr       EXIT SUB
1519 Ss2    anwenderinit:
1520 kT4      OPEN "ANWENDER" FOR OUTPUT AS 1
1521 uD       PRINT #1,"0"
1522 M6       CLOSE 1
1523 bd       END SUB
1524 ZM0  REM  Neuen Anwender einfuegen
1525 yt2    neuanwender:
1526 ep4      IF man=10 THEN man=0:RETURN
1527 hL       man=man+1:anw=man
1528 I0       na$(man)=z$
1529 zi       FOR s=0 TO 7
1530 rR6        FOR t=0 TO 2
1531 JB8          fa(man,s,t)=farb(s,t)
1532 ns6        NEXT
1533 ot4      NEXT
1534 F7       CHDIR "DF0:"
1535 zi       OPEN "ANWENDER" FOR OUTPUT AS 1
1536 f9       PRINT #1,man
1537 GB       FOR s=1 TO man
1538 IP6        PRINT #1,na$(s)
1539 Fu         FOR t=0 TO 7
1540 7d8          FOR u=0 TO 2
1541 DqA            PRINT #1,fa(s,t,u)
1542 x28          NEXT
1543 y36        NEXT
1544 z44      NEXT
1545 jT       CLOSE 1
1546 P5       OPEN "DATEIEN."+LEFT$(na$(man),3) FOR OUTPUT AS 1
1547 Kd       PRINT #1,"0"
1548 mW       CLOSE 1
1549 Rd       OPEN "BENUTZT."+LEFT$(na$(man),3) FOR OUTPUT AS 1
1550 Ng       PRINT #1,"0"
1551 pZ       CLOSE 1
1552 00       RETURN
1553 XN0  REM  Anwenderdatei speichern
1554 Tk2    faspeichern:
1555 J24      OPEN "ANWENDER" FOR OUTPUT AS 1
1556 zT       PRINT #1,man
1557 aV       FOR s=1 TO man
1558 cj6        PRINT #1,na$(s)
1559 ZE         FOR t=0 TO 7
1560 Rx8          FOR u=0 TO 2
1561 XAA            PRINT #1,fa(s,t,u)
1562 HM8          NEXT
1563 IN6        NEXT
1564 J04      NEXT
1565 3n       CLOSE 1
1566 cE       RETURN
1567 Gt0  REM  Benutzungsdaten laden
1568 w22    anwbenutzung:
1569 Tg4      OPEN "BENUTZT."+LEFT$(na$(anw),3) FOR INPUT AS 1
1570 Gr       INPUT #1,mabn
1571 05       FOR s=1 TO mabn
1572 X76        FOR t=0 TO 2
1573 hY8          INPUT #1,abn$(s,t)
1574 TY6        NEXT
1575 UZ4      NEXT
1576 Ey       CLOSE 1
1577 nP       RETURN
1578 9B0  REM  Uhrzeit ausgeben
1579 5M2    uhr:
1580 Vr4      COLOR 3,2
1581 Qw       pixlocate 550,16:PRINT TIME$
1582 sU       RETURN
1583 qe0  REM  Datas
1584 vA2    DATA Sonntag,Montag,Dienstag,Mittwoch,Donnerstag,Freita
            g,Samstag
1585 xC     DATA Vokabel - Trainer  Vers. 1.0
1586 J3     DATA Diskettenbetrieb
1587 gG     DATA Vokabeln eingeben
1588 w3     DATA Vokabeln ausgeben
1589 mb     DATA Vokabeln editieren
1590 AJ     DATA Vokabeln suchen
1591 c0     DATA Vokabeln abfragen
1592 AU     DATA Statistik ausgeben
1593 CZ     DATA Programm beenden
1594 2d     DATA Sonderfunktionen
1595 wQ     DATA Vokabeldatei laden
1596 OD     DATA Vokabeldatei speichern
1597 Nb     DATA"Vokabeldatei löschen"
1598 gx     DATA Diskette wechseln
1599 4R     DATA Im Speicher suchen
1600 1Z     DATA Auf Diskette suchen
1601 kb     DATA"Vokabeln einprägen"
1602 fl     DATA"Übersetzung abfragen"
1603 81     DATA Fremdwort abfragen
1604 JS     DATA"Schrift üben"
1605 OK     DATA Abfrage in Reihenfolge
1606 qg     DATA Abfrage gemischt
1607 JU     DATA Abfrage nach "Können"
1608 U7     DATA"Statistik von Übersetzung"
1609 45     DATA Statistik von Fremdwort
1610 Al     DATA"Statistik von Schrift-Üben"
1611 1F     DATA
1612 mG     DATA
1613 nH     DATA
1614 oI     DATA
1615 kg     DATA Auf der eingelegten Diskette*sind keine Vokabeldat
            eien.
1616 v0     DATA Es sind keine Vokabeln zum*Sichern im Speicher.
1617 Jq     DATA"Der für Vokabeln reservierte*Speicherbereich ist v
            oll."
1618 oi     DATA"Diese Vokabeldatei ist nicht auf der*eingelegten D
            iskette.*Bitte Diskette überprüfen."
1619 Ng     DATA Die Vokabeln im Speicher sind*noch nicht gespeiche
            rt worden.*Soll dennoch eine andere Datei*geladen werden
             ?
1620 Lv     DATA Mit diesem Dateinamen existiert*schon eine andere
            Datei.
1621 fo     DATA Soll die Vokabeldatei wirklich*gelöscht werden ?
1622 VD     DATA"Auf dieser Diskette sind keine*weiteren Eintragung
            en möglich."
1623 8Y     DATA Die Vokabeln im Speicher sind*noch nicht gespeiche
            rt worden.*Soll die Diskette dennoch*gewechselt werden ?
1624 4w     DATA Bitte die Diskette wechseln.
1625 5s     DATA Auf der eingelegten Diskette befinden*sich keine V
            okabeldateien vom angege-*benen Anwender.
1626 ut     DATA Es sind keine Vokabeln zum*Ausgeben im Speicher.
1627 9S     DATA Es sind keine Vokabeln zum*Editieren im Speicher.
1628 yw     DATA Vokabeln verbessern
1629 mm     DATA"Vokabeln löschen"
1630 73     DATA"Speicher löschen"
1631 ti     DATA"Soll die Vokabel wirklich*gelöscht werden ?"
1632 JQ     DATA"Die bearbeitete Vokabeldatei ist*noch nicht gespei
            chert worden.*Soll der Speicher dennoch*gelöscht werden
            ?"
1633 KA     DATA"Soll wirklich der ganze*Speicher gelöscht werden ?
            "
1634 yG     DATA Es sind keine Vokabeln zum*Suchen im Speicher.
1635 xh     DATA Ist dies die gesuchte*Vokabel ?
1636 tn     DATA Die gesuchte Vokabel*ist nicht im Speicher.
1637 x8     DATA"Zurück zum Hauptmenü"
1638 nV     DATA Es sind keine Vokabeln zum*Abfragen im Speicher.
1639 4z     DATA"Es sind keine Vokabeln im Speicher,*zu denen eine
            Statistik ausgegeben*werden könnte."
1640 Lp     DATA Die bearbeitete Vokabeldatei ist*noch nicht gespei
            chert worden.*Soll das Programm dennoch*abgebrochen werd
            en ?
1641 qw     DATA Soll das Programm wirklich*beenden werden ?
1642 Gk     DATA
1643 Wm     DATA"Farbwerte ändern"
1644 21     DATA Info zu Abfragedaten
1645 aJ     DATA Benutzung des Programms
1646 t0     DATA"Es sind keine Vokabeln im Speicher,*zu denen die A
            bfragedaten ausgege-*ben werden könnten."
1647 xl     DATA Die Vokabeldatei ist bisher*nicht in dieser Form g
            e-*speichert worden.
1648 yq     DATA Die aktuelle Vokabeldatei*wurde noch nicht abgefra
            gt.
1649 nw     DATA Der aktuelle Anwendername*ist unbekannt.
1650 db     DATA -1
1651 OT     DATA 232,233,234,224,226,217,219,244,231,238
1652 2v     DATA 0,0.04,0.1,0.16,0.22,0.29,0.35,0.41,0.47,0.54,0.6,
            0.66,0.72,0.79,0.85,0.91
1653 W0     DATA 10,9,7,10,9,7,13,13,13,1,1,1,16,1,1,12,10,3,1,10,1
            ,1,1,15
(C) 1988 M&T
```

Listing 2. (Schluß)

EINGABEHILFE

Checksummer

Dieses Programm ist unentbehrlich beim Abtippen unserer Listings. Es hilft, Tippfehler zu vermeiden und spart viel Zeit.

Ein längeres Listing ohne Fehler abzutippen ist (fast) unmöglich. Aus diesem Grund haben wir in Ausgabe 3/88 des AMIGA-Magazins eine Eingabehilfe — den Checksummer »Checkie 42« veröffentlicht. Die hier vorgestellte Version 1.1 enthält erweiterte Funktionen und bietet mehr Komfort. Damit möglichst viele unserer Leser dieses Programm auch tatsächlich anwenden, haben wir es möglichst kurz gehalten und in einer Sprache programmiert, die alle Abtipper besitzen: Amiga-Basic.

Die Form der Listings

Die Listingzeilen bestehen aus einer bis zu vierstelligen Zeilennummer, der zwei- beziehungsweise dreistelligen Prüfsumme und der eigentlichen Programmzeile. Beispiel:

```
10 TTO print "Hallo!"
|   └──Prüfcode
└──────Zeilennummer
```

Nach einer Leerstelle im Anschluß an die Zeilennummer stehen bis zu drei Zeichen Prüfcode. Die einzelnen Zeichen können sein eine Ziffer (»0« bis »9«), ein kleiner Buchstabe (»a« bis »z«) oder ein Großbuchstabe (»A« bis »Z«). Die ersten beiden Zeichen der Prüfsumme sind der eigentliche Prüfcode. Im dritten Zeichen ist die Spaltenposition der ersten »Nicht-Leerstelle« verschlüsselt. Das ist für diejenigen Anwender interessant, welche die Struktur des Listings, also die Einrückungen durch Leerzeichen, übernehmen wollen. Ist dies nicht Ihre Absicht, können Sie die Eingabe der Checksumme schon nach den ersten beiden Zeichen mit <Return> abschließen. Bei Checkie 42 muß die Groß- und Kleinschreibung so wie im Listing abgedruckt übernommen werden.

> **Eingabehinweis:**
> Geben Sie »Checkie 42« (Version 1.1) bitte mit Checkie 42 aus der AMIGA-Ausgabe 3/88 oder 12/87 ein. Sollten Sie die alte Eingabehilfe nicht besitzen, so tippen Sie das Listing für die Version 1.1 im normalen Basic-Editor ohne Prüfsummen und Zeilennummern ab.

Der Umgang mit Checkie 42

Nach dem Start fragt das Programm nach einem Dateinamen. Unter dem angegebenen Namen speichert Checkie 42 die eingegebenen Listingzeilen ab. Existiert bereits eine Datei mit diesem Namen auf der Diskette, so haben Sie mit der Abfrage »Nur Checksummer ausgeben?« zwei Möglichkeiten:

<j> Ausgabe der Datei mit Checksumme auf den Bildschirm oder den Drucker.

<n> Einlesen der Programmzeilen aus der vorhandenen Datei und Eingabe der Checksumme mit der Tastatur.

Beide Alternativen sind gedacht für Anwender, die ein Listing nicht mit dem Zeileneditor des Checkie, sondern mit einem schnelleren und/oder komfortableren Editor ihrer Wahl — zum Beispiel dem Editor von Amiga-Basic (mit »..",a« speichern) erfaßt haben.

Checkie 42 errechnet nach der Eingabe <j> die Prüfsummen Ihres Textes und Sie können diese dann mit dem Listing im AMIGA-Magazin vergleichen. Bei der Ausgabe auf den Bildschirm schreibt das Programm die Programmzeilen inklusive Checksummen zusätzlich in eine Datei auf Diskette mit dem Zusatz ».chk«. Diese können Sie später zum Beipiel mit dem CLI-Befehl TYPE ohne erneute Berechnung der Prüfsumme noch einmal ausgeben.

Haben Sie »Nur Checksumme ausgegeben?« mit »n« beantwortet, dann können Sie dem Programm den Vergleich überlassen, in dem die Frage »Eingabe aus Datei« mit »j« beantwortet wird. Dann brauchen Sie nur noch die Checksummen eingeben. Der Checksummer holt sich die Zeile aus der angegebenen Datei statt von der Tastatur. Entspricht die eingegebene Prüfzahl nicht der errechneten, kann die Zeile gleich korrigiert werden.

Beantworten Sie obige Frage mit »n«, zählt Checkie die in der Datei vorhandenen Zeilen und wartet mit der Zeilennummer »Anzahl+1« auf die Eingabe einer neuen Zeile. Alle weiteren Eingaben hängt das Programm an die bestehende Datei an. Diese Funktion ist sinnvoll, wenn Sie ein Listing in mehreren Teilen abtippen wollen.

So tippen Sie Listings ab

Haben Sie anfangs einen Dateinamen eingegeben oder Sie hängen neue Zeilen an eine bestehende Datei an, dann arbeiten Sie im normalen Eingabemodus. Checkie 42 schlägt dabei eine Zeilennummer vor und wartet auf die Prüfsumme. Nach Eingabe derselben taucht der Cursor zwischen den zwei Trennstrichen auf. Dort muß nun die Zeile »ohne« Zeilennumer und Prüfsumme eingegeben werden. Nach Betätigen der Taste <Return> berechnet Checkie die Prüfsumme. Leerstellen vor und hinter der Programmanweisung werden ignoriert. Stimmen Programmzeile und Prüfsumme mit derjenigen im Listing überein, speichert der Checksummer die Eingabe ab und wartet auf die nächste Zeile.

Einfügemodus: Wahrscheinlich wird eine abgetippte Zeile mal einen Fehler enthalten. Checkie 42 positioniert den Cursor dann an den Anfang der Zeile und wartet auf die korrekte Eingabe. Korrekturen lassen sich mit der Backspace- oder Delete-Taste durchführen. Um Zeichenfolgen einzufügen, kann kurzfristig mit <F2> der Einfügemodus eingeschaltet werden. Dieser Modus sollte allerdings nach der Fehlerkorrektur wieder ausgeschaltet werden, da er die Eingabe verlangsamt.

Sonderfall-Prüfsumme ignorieren: Möchten Sie zum Beispiel eine Kommentarzeile nicht »original« übernehmen, läßt sich trotz einer falschen Prüfsumme eine Übernahme der Zeile mit der Funktionstaste <F6> erzwingen. Sie können damit aber auch falsche Programmzeilen übernehmen. Verwenden Sie deshalb die Taste <F6> nicht gewohnheitsmäßig. Der Checksummer teilt Ihnen nach Beenden des Programms mit, wieviel Zeilen er ungeprüft übernommen hat.

Prüfsumme und Zeilennummer ändern: Natürlich kann es auch vorkommen, daß die Programmzeile zwar richtig abgetippt wurde, sich bei der Prüfsumme aber ein Fehler eingeschlichen hat. Nach Betätigen von <F1> kann die Prüfsumme korrigiert werden. Während der Eingabe der Prüfsumme läßt sich mit <F7> die vom Programm vorgeschlagene Zeilennummer verändern. Damit können Sie gezielt nur bestimmte Teile eines Listings übernehmen.

Haben Sie eine mit einem anderen Editor geschriebene Programmdatei überprüft und nur in wenigen Zeilen Fehler festgestellt, lassen sich durch Vorgabe der Nummern diese Zeilen gezielt ändern. Bei Angabe der Zeilennummer in aufsteigender Reihenfolge benötigt das Programm übrigens erheblich weniger Zeit für die Suche der Zeilen in der jeweiligen Datei. Um die versehentliche Übernahme fehlerhafter Zeilen zu verhindern, sperrt das Programm bei fehlender Übereinstimmung der Prüfsummen die Taste <F7> (Änderung der Zeilennummer).

Fehlerfrei abtippen

Eingabe beenden: Die Kombination <Ctrl-e> beendet den Programmlauf nach vollständiger Eingabe des Listings oder für eine Unterbrechung.

Am Schluß noch ein Tip für alle Leser, denen unser Basic-Editor zu langsam ist. Die Berechnung der Prüfsummen erfolgt im Unterprogramm »CalcSumme«. Dieser Teil ist sehr einfach in schnellere Sprachen, wie beispielsweise C, umsetzbar.

Wer schon einmal Fehler in einem abgetippten Listing gesucht hat, der weiß, wie frustrierend diese Arbeit sein kann. Nutzen Sie deshalb den »Checkie 42«. Sie sparen viel Zeit und müssen sich nicht dauernd auf die Suche nach tückischen Fehlern begeben.

(Dieter Behlich/kn)

```
CqO REM ***********************************
KF  REM *** Checksummer: Checkie 42       ***
4c  REM ***         Version 1.1           ***
Ft  REM ***********************************
Z7  Start:
W42   GOSUB Init
Pn    GOSUB OpenDatei
mb    IF dn$="" THEN Ende
b2    GOSUB Bild
OwO NeueZeile:
K42   GOSUB loeschen
zT    GOSUB EingabeSumme
pt    IF FEnde=wahr THEN Ende
MVO Wiederholung:
sQ2   GOSUB EingabeZeile
sw    IF FEnde=wahr THEN Ende
mO    GOSUB CalcSumme
fB    IF FZok = falsch THEN GOTO Wiederholung
oC    GOSUB Uebernahme
iv    GOTO NeueZeile
mAO Ende:
7S2   GOSUB fertig
72    END
UGO Init:
fD2   wahr=-1
fy    falsch=0
Kh    LZeile=240  : REM Anzahl Zeichen/Zeile
tz    LBZeile=60  : REM Anzahl Zeichen/Bildschirmzeile
OG    AnzBZeilen=LZeile/LBZeile
N1    zx=6 : zy=14   : REM Position Zeile
PC    sx=20 : sy=5   : REM Position Checksumme
mF    AnzCsZ=3       : REM Anzahl Ziffern/Checksumme
QF    DIM z(LZeile)
05    DIM cs(AnzCsZ*2)
iz    cs(AnzCsZ)=0   : REM Zeilenstart
gv    a=0 : b=0 : c=0  : REM Hilfsvariablen
AI    i=0 : j=0 : k=0  : REM Zählvariablen
XL    FZok = wahr : FCz3=100
Wn    Checkfile=0  : Zeile=1
Bb    READ Faktor(i)
X1    WHILE Faktor(i)<>0
ga4     i=i+1
cm      READ Faktor(i)  : REM Faktorenreihe
9x2   WEND
Gm    AnzFak=i
RB    DATA 2,3,4,5,6,0
7j    RETURN
PWO OpenDatei:
BR2   ON ERROR GOTO Dateifehler
MS    CLS
Vu    LOCATE 3,5:dn$="":INPUT "Dateiname: ";dn$
Fp    OPEN dn$ FOR INPUT AS #1
KU    IF FFehler=53 THEN NeueDatei
4s    LOCATE 5,5:PRINT "Nur Checksummen ausgeben? (j/n) ":e$=INP
      UT$(1)
gM    IF e$="j" THEN
Kq4     LOCATE 7,5:e$="":PRINT "Bildschirm oder Drucker? (b/d) "
        :e$=INPUT$(1)
Ku      IF e$="d" THEN
KM7       OPEN "prt:" FOR OUTPUT AS #4
YH4     ELSE
7Y6       OPEN dn$+".chk" FOR OUTPUT AS #4
vo4     END IF
RR      WHILE NOT EOF(1)
yj6       LINE INPUT#1,e$
fg        apos=LEN(e$)
to        z(apos)=32
4L        FOR i=1 TO apos
sq8         z(i-1)=ASC(MID$(e$,i,1))
Xn6       NEXT i
cE        GOSUB CalcSumme
vp        i=0
Rh        WHILE z(i)=32 : i=i+1 : WEND
4v        IF FCz3=i THEN
f98         cs(6)=-16     : REM Leerstelle
nW6       ELSE
XA8         cs(6)=i : FCz3=i
A36       END IF
2o        PRINT USING "####";Zeile; : PRINT " ";
Wf        PRINT#4,USING "####";Zeile; : PRINT#4," ";
mC        FOR i=4 TO 6
7d8         IF cs(i)>35 THEN cs(i)=cs(i)+6
nS          IF cs(i)>9 THEN cs(i)=cs(i)+7
Jy          PRINT CHR$(cs(i)+48);
8L          PRINT#4,CHR$(cs(i)+48);
n36       NEXT i
19        PRINT " ";e$
tl        PRINT#4," ";e$
Qm        Zeile=Zeile+1
rf4     WEND
IF      CLOSE 1 : CLOSE 4
H4      dn$=""
Fv      PRINT:PRINT "Taste drücken"
5r      e$=INPUT$(1)
rT      RETURN
SL2   END IF
70    LOCATE 7,5:PRINT"Eingabe aus Datei? (j/n) ":e$=INPUT$(1)
Li    IF e$="j" THEN
u24     Checkfile=1
08      CLOSE 1
eX      GOSUB backup
WZ      OPEN dn$+".bak" FOR INPUT AS #1
Sk      OPEN dn$ FOR OUTPUT AS #2
Oc      RETURN
bU2   END IF
77    WHILE NOT EOF(1)
eP4     LINE INPUT#1,e$
j5      Zeile=Zeile+1
Ay2   WEND
YI    CLOSE 1
8AO NeueDatei:
aM2   OPEN dn$ FOR APPEND AS #2
91    RETURN
7QO Dateifehler:
Og2   FFehler=ERR
PB    IF FFehler=53 THEN RESUME NEXT
vy    ON ERROR GOTO 0
aDO backup:
yY2   OPEN dn$ FOR INPUT AS#1
SJ    OPEN dn$+".bak" FOR OUTPUT AS #2
8F    flen=LOF(1)
ih    WHILE flen>=32000
i14     e$=INPUT$(32000,1)
tL      PRINT#2,e$;
QE2   WEND
e7    IF flen THEN
SB4     e$=INPUT$(flen,1)
xP      PRINT#2,e$;
zs2   END IF
zg    CLOSE 2  : CLOSE 1
R3    RETURN
dOO Bild:
f12   CLS
Gi    LOCATE 2,10
yY    PRINT ">>>>>>> Checkie 42 - Der ultimative Checksum
      mer! <<<<<<<"
xR    LOCATE 13,zx
Le    FOR i=1 TO 2
kx4     FOR j=1 TO LBZeile
HR6       PRINT "-";
hy4     NEXT j
H1      LOCATE 13+AnzBZeilen+1,zx
hx2   NEXT i
Hi    LOCATE 5,30:PRINT "<CTRL e> = Programm beenden"
vd    LOCATE 7,8:PRINT "<F2> = Einfügemodus ein"
uQ    LOCATE 7,40:PRINT "<F1> = Checksumme ändern"
1H    LOCATE 9,40:PRINT "<F6> = Zeile speichern"
ME    LOCATE 9,8:PRINT "<F3> = Zeile löschen"
v5    LOCATE 11,8:PRINT "<F7> = Zeilennummer ändern"
jL    RETURN
5fO loeschen:
w82   LOCATE zy,1
8S    FOR i=1 TO AnzBZeilen
jf4     PRINT TAB(zx);SPACE$(LBZeile)
t92   NEXT i
OO    FOR i=0 TO LZeile
V84     z(i)=32
wC2   NEXT i
3F    LOCATE zy,1
om    PRINT USING "####";Zeile
dy    apos=0
vX    RETURN
v5O EingabeSumme:
va2   Farbe=0 : i=1
AB    LOCATE sy,sx+1
Xm    PRINT "..."
O4O blinken:
aC2   Farbe=Farbe XOR 1
QH    COLOR Farbe,0
P6    LOCATE sy,sx-15
1n    PRINT "Checksumme:"
ni    LOCATE sy,sx+i
ya    e$=INKEY$
1T    IF e$="" THEN blinken
Xp    COLOR 1,0
dl    e=ASC(e$)
fM    IF e=5 THEN
KU4     FEnde=wahr   : REM <CTRL e>
hf      i=AnzCsZ+1   : REM fertig
Cc2   ELSEIF e=135 THEN
dc4     GOSUB NeuZeile
Bz      e=135
```

```
LB2   ELSEIF e=8 THEN
4h4     IF i>1 THEN
sZ6       i=i-1       : REM <BS>
XY        LOCATE sy,sx+i : PRINT "."
un4     END IF
Tn2   ELSEIF e=13 THEN
KZ4     IF i=AnzCsZ THEN i=AnzCsZ+1 : REM <CR>
cL2   ELSE
ra4     IF e>47 AND e<58 THEN
vi6       e=e-48                     : REM 0-9
AI4     ELSEIF e>64 AND e<91 THEN
Rv6       e=e-55                     : REM A-Z
jX4     ELSEIF e>96 AND e<123 THEN
Zx6       e=e-61                     : REM a-z
jS4     ELSE
yz6       GOTO blinken               : REM weder noch
6z4     END IF
JY      PRINT e$;
ve      cs(i)=e
D7      i=i+1
A32   END IF
wn    IF i<=AnzCsZ THEN blinken
f50 ESEnde:
2K2   COLOR 1,0
0h    LOCATE sy,sx-15
c0    PRINT "Checksumme:"
gI    RETURN
aE0 NeuZeile:
dU2   IF FZok = wahr THEN
P44     NeuZeile=0
nn      WHILE e<>13 OR NeuZeile=0
Ux6       LOCATE zy,1:PRINT USING "####";NeuZeile;
oF        e=ASC(INPUT$(1))
gP        IF e>47 AND e<58 THEN NeuZeile=NeuZeile*10+e-48
Jn        IF NeuZeile > 9999 THEN e=8
in        IF e=8 THEN NeuZeile=INT(NeuZeile/10)
vj4     WEND
kh      IF Checkfile THEN
tQ6       IF NeuZeile < Zeile THEN
yy8         WHILE NOT EOF(1)
VGA           LINE INPUT#1,e$
cd            PRINT#2,e$
1p8         WEND
MH          CLOSE 1 : CLOSE 2
fY          GOSUB backup
Xa          OPEN dn$+".bak" FOR INPUT AS #1
T1          OPEN dn$ FOR OUTPUT AS #2
9t          Zeile=1
cV6       END IF
Yp        WHILE (NeuZeile > Zeile) AND (NOT EOF(1))
fQ8         LINE INPUT#1,e$
mn          PRINT#2,e$
17          Zeile=Zeile+1
C06       WEND
cR        IF EOF(1) THEN
bL8         CLOSE 1
8i          NeuZeile=Zeile
uN          LOCATE zy,1:PRINT USING "####";NeuZeile;
9G          Checkfile=0
ng6       END IF
oh4     END IF
vV      Zeile=NeuZeile
qj2   END IF
Ht0 RETURN
1H  EingabeZeile:
n72   x=cs(AnzCsZ)
BW0 weiter:
ga2   cy=zy+INT(x/LBZeile):cx=zx+(x MOD LBZeile)
vy    LOCATE cy,cx
o6    COLOR 0,1
mI    PRINT CHR$(z(x));
y1    LOCATE cy,cx
68    IF x>apos THEN apos=x
xx    IF Checkfile AND FZok THEN
w14     IF EOF(1) THEN
0R6       Checkfile=0 : CLOSE 1
jS4     ELSE
ix6       e$=INPUT$(1,1)
6z4     END IF
mV2   ELSE
Q24     e$=INKEY$
922   END IF
gp4     IF e$="" THEN weiter
OI      COLOR 1,0
OW      PRINT CHR$(z(x));
CF      LOCATE cy,cx
8G      e=ASC(e$)
UR2   IF ((e AND 127)<32) OR e=127 THEN Controlcode
op    IF imode THEN GOSUB insert
I6    PRINT e$
Ki    z(x)=e : e=30
3D0 Controlcode:
gF2   IF e=13 OR e=10 THEN
1N4     RETURN
092   ELSEIF e=30 THEN
2p4     a=1
Ea2   ELSEIF e=29 THEN
034     a=LBZeile
6G2   ELSEIF e=31 THEN
DD4     a=-1
Gb2   ELSEIF e=28 THEN
Rg4     a=-LBZeile
9s2   ELSE
UX4     GOTO noCrs
WP2   END IF
hz    x=x+a
5Z    IF x>=0 AND x<LZeile THEN weiter
tD    x=x-a
HB    GOTO weiter
kE0 noCrs:
dN2   IF e=8 THEN
1s4     IF x>0 THEN
Kk6       x=x-1
AM        LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
Bb        FOR i=x TO apos
788         z(i)=z(i+1)
Ja          PRINT CHR$(z(i));
Z0          IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
FV6       NEXT i
pr        z(apos)=32 : PRINT " "
yR        apos=apos-1
ng4     END IF
H12   ELSEIF e=127 THEN
Kk4     FOR i=x TO apos
GH6       z(i)=z(i+1)
SJ        PRINT CHR$(z(i));
i9        IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
0e4     NEXT i
y0      z(apos)=32 : PRINT " "
7a      apos=apos-1
Vy2   ELSEIF e=129 THEN
oI4     GOSUB EingabeSumme
sC      x=cs(AnzCsZ)
9U2   ELSEIF e=130 THEN
513    imode=imode XOR 1
tU5      LOCATE 7,28
yZ3    IF imode THEN
iT5      PRINT "aus"
jS3    ELSE
YV5      PRINT "ein"
6z3    END IF
Kg2   ELSEIF e=131 THEN
L54     GOSUB loeschen
3N      x=cs(AnzCsZ)
7Y2   ELSEIF e=134  THEN
bD4     RETURN
PR2   ELSEIF e=5 THEN
ZT4     FEnde=wahr
eG      RETURN
F82   END IF
xr    GOTO weiter
If0 insert:
oJ2   IF apos>x THEN
3o4     FOR i=apos TO x STEP -1
Qt6       z(i+1)=z(i)
q64     NEXT i
PH      z(x)=32
Ny      apos=apos+1
Kp      IF apos=LZeile THEN apos=apos-1:z(LZeile)=32
uK      FOR i=x TO apos
1I6       PRINT CHR$(z(i));
Hi        IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
xD4     NEXT i
yA      LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
UN2   END IF
vX    RETURN
cX0 CalcSumme:
3s2   a=0 : b=0 : c=0
z2    IF e=134 THEN
pv4     FZok=wahr
WX      FF6=FF6+1
Gz2   ELSE
hv4     WHILE z(apos)=32 AND apos>0
pI6       apos=apos-1
9x4     WEND
N1      IF apos>0 THEN
036       WHILE z(c)=32
```

Listing. Der verbesserte »Checkie 42«. Bitte mit der ersten Version von »Checkie 42« oder ohne Prüfsummen und Zeilennummern eingeben.

```
357 Bt8         c=c+1
358 D16       WEND
359 jc4     END IF
360 EJ      FOR i=c TO apos
361 Y26        j=(i-c) MOD AnzFak
362 Qs         k=(i+1-c) MOD AnzFak
363 s1         a=a+((z(i) AND 127)-32)*Faktor(j)
364 2E         b=b+((z(i) AND 127)-32)*Faktor(k)
365 Ka4     NEXT i
366 pA      cs(4)=a+Zeile-INT((a+Zeile)/62)*62
367 4N      cs(5)=b+Zeile-INT((b+Zeile)/62)*62
368 4Q      FZok=(cs(1)=cs(4)) AND (cs(2)=cs(5))
369 tm2   END IF
370 Kw    RETURN
371 pM0 Uebernahme:
372 xD2   FOR i=0 TO apos
373 ih4     PRINT#2,CHR$(z(i));
374 TJ2   NEXT i
375 VZ4     PRINT#2,""
376 5R2   Zeile=Zeile+1
377 R3    RETURN
378 N90 fertig:
379 MJ2   IF Checkfile THEN
380 ZZ4     WHILE NOT EOF(1)
381 6r6       LINE INPUT#1,e$
382 DE        PRINT#2,e$
383 cQ4     WEND
384 Ok      CLOSE 1
385 922   END IF
386 5q    CLOSE 2
387 nt    CLS
388 Rg    LOCATE 12,35
389 WL    PRINT "F E R T I G !!!"
390 I9    LOCATE 20,1
391 D1    IF FF6<>0 THEN
392 C14     PRINT "ACHTUNG!!! ";
393 9H      PRINT FF6;" Zeile(n) wurde(n) ungeprüft gespeichert."
394 IB2   END IF
395 jL    RETURN
(C) 1988 M&T
```

Listing. Der verbesserte »Checkie 42« (Schluß)

Inserenten-verzeichnis

Impressum

Herausgeber: *Carl-Franz von Quadt, Otmar Weber*

Chefredakteur: *Albert Absmeier*
Leitender Redakteur: *Gottfried Knechtel – verantwortlich für den redaktionellen Teil*
Chef vom Dienst: *Susanne Kirmaier*
Redaktion: *Klaus Schrödl, Ralf Sablowski*
Redaktionsassistenz: *Andrea Kaltenhauser, Brigitte Bobenstetter, Helga Weber (202)*
Mitarbeiter der Redaktion: *Martin Jobst, Andreas Lietz, Christian Wolff*

Alle Artikel sind mit dem Kennzeichen des Redakteurs (kn = Gottfried Knechtel, sk = Klaus Schrödl, rs = Ralf Sablowski) und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet

Art-director: *Friedemann Porscha*
Layout: *Erich Schulze (Cheflayouter), Bernd Wiehl*
Fotografie: *Sabine Tennstaedt, Claus Troendle, Didi Zille*
Titelgestaltung: *Friedemann Porscha*
Spritzgrafik: *Ewald Standke*
Computergrafik: *Werner Nienstedt*

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-415656, Telex: 862329 mut ch
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351
Österreich: Markt&Technik Ges. mbH Hermann Raniger, Große Neugasse 28, A 1040-Wien, Tel. 0043-222-8579455, Telex: 047-132532

Manuskripteinsendungen: *Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen und dazu, daß Markt&Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.*

Produktionsleiter: *Klaus Buck (180)*

Anzeigenverkaufsleitung: *»Populäre Computerzeitschriften«: Alexander Narings (780)*
Anzeigenleitung: *Alicia Clees (313) – verantwortlich für Anzeigen*

Anzeigenformate: *1/1 Seite ist 266 Millimeter hoch und 185 Millimeter breit (2 Spalten à 86 Millimeter oder 4 Spalten à 43 Millimeter). Vollformat 297x210 Millimeter.*

Anzeigenverwaltung und Disposition: *Lisa Landthaler (233)*

Anzeigen-Auslandsvertretung: England: *F. A. Smyth&Associates Limited, 23a, Aylmer Parade, London, N2 OPQ. Telefon:0044/1/3405058, Telefax: 0044/1/3419602*
Taiwan: *Third Wave Publishing Corp., 1-4 Fl. 977 Min Shen E. Road, Taipei 10581, Taiwan, R.O.C., Tel. 00886/2/7630052, Telefax: 00886/2/7658767, Telex: 078529335*

Vertriebsleiter: *Helmut Grünfeldt (189)*

Leiter Vertriebs-Marketing: *Benno Gaab (740)*

Vertrieb Handelsauflage: *Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1,*

Bezugsmöglichkeiten: *Leser-Service: Telefon (089) 4613-366. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.*

Preis: *Das Einzelheft kostet DM 16,–*

Druck: *SOV Graphische Betriebe, Laubanger 23, 8600 Bamberg*

Redaktionsdirektor: *Michael M. Pauly*

Vorstand: *Otmar Weber (Vors.), Bernd Balzer, Werner Brodt*

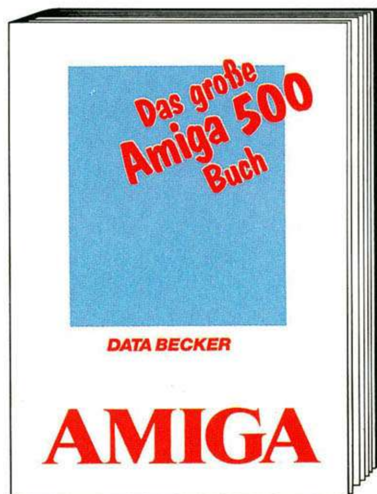
Leiter Unternehmensbereich »Populäre Computerzeitschriften«:
Eduard Heilmayr, Werner Pest

Redaktionskoordination »Populäre Computerzeitschriften«:
Hans-Günther Beer

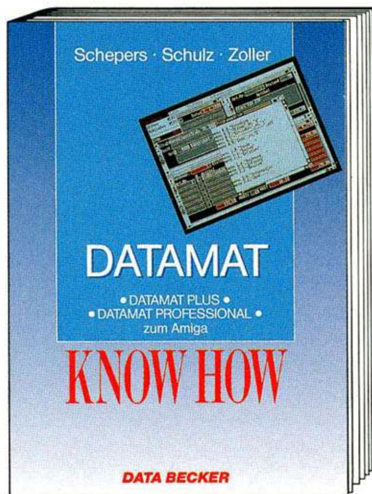
Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen: *Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0, Telex 5-22052*

Telefon-Durchwahl im Verlag: *Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/4613 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.*

AMIGA BUCHHITS



Alles über Ihre kleine Freundin.
Wie gut das Handbuch auch sein mag, das große Amiga-500-Buch macht sich durch komplettes Detailwissen einfach unentbehrlich. Ob zur Hardware, zur Workbench oder zur Programmierung – hier finden Sie das Know-how, das einen Profi auszeichnet. Doch auch wenn Sie dieses Buch komplett durchgearbeitet haben, bleibt Ihnen das große Amiga-500-Buch als nützliches und zuverlässiges Nachschlagewerk erhalten. Ein weiterer Grund, es immer griffbereit neben dem Amiga stehen zu haben. Das große Amiga-500-Buch – die Pflichtlektüre für jeden Amiga-500-Anwender.
Das große Amiga-500-Buch
Hardcover, ca. 450 Seiten, DM 49,–
erscheint ca. 11 / 88



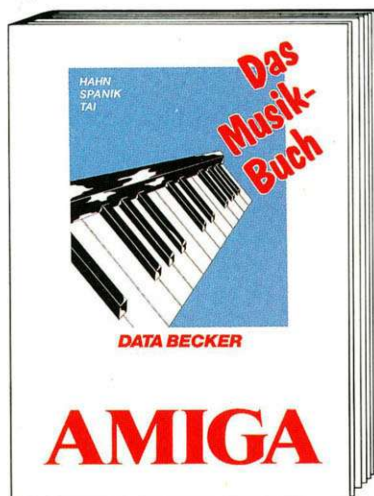
Das Know-how zu Ihrem DATAMAT-Programm.
Das Programm nach Maß: DATAMAT. In drei verschiedenen Versionen ist dieses Programm für den Amiga erhältlich. Als reine Dateiverwaltung, als einfache Datenbank und als Datenbank mit integrierter Programmiersprache. Wo liegen die Unterschiede der einzelnen Programme, was leisten sie und vor allem, wie setzt man sie optimal für eigene Anwendungen ein? Die Antworten finden Sie in „DATAMAT Know-how". Zahlreiche Tips & Tricks runden das Ganze ab. DATAMAT Know-how – damit die Programme auch halten, was sie versprechen.
DATAMAT Know-how
ca. 400 Seiten, DM 39,–
erscheint ca. 11 / 88



Rund um die Datenbank Superbase.
Ob Superbase Personal II oder Superbase Professional – das große Superbase-Buch zeigt Ihnen, was diese Datenbanken im einzelnen leisten. Angefangen von der Dateidefinition, über die Dialogboxen und Schaltsymbole bis hin zur mächtigen Programmiersprache DML finden Sie hier alles, um Ihr Programm optimal für eigene Anwendungen nutzen zu können. Wie immer Sie Superbase auch einsetzen mögen, privat oder gewerblich, mit diesem Buch machen Sie mehr daraus. Das große Superbase-Buch – und Sie lernen Ihr Programm so richtig kennen.
Das große Superbase-Buch
ca. 350 Seiten, DM 39,–
erscheint ca. 11 / 88



Perfekte Texte mit WordPerfect.
WordPerfect bietet eine unglaubliche Anzahl von Funktionen – da muß man schon bestens Bescheid wissen, um dieses Programm bis zum letzten nutzen zu können. Das große Buch zu WordPerfect kann Ihnen dabei helfen. Es ermöglicht Ihnen nicht nur einen schnellen Start, sondern vermittelt Ihnen auch jene Detailkenntnisse, die Sie brauchen, um Ihre Texte perfekt zu gestalten. Natürlich auch in diesem Buch: zahlreiche Tips & Tricks für Ihre tägliche, praktische Arbeit. Für buchstäbliche Vielschreiber einfach ein Muß.
Das große Buch zu WordPerfect.
Hardcover, ca. 320 Seiten, DM 39,–
erscheint ca. 11 / 88



Der Ton macht die Musik.
Zaubern Sie zarte Klänge oder heiße Rhythmen aus Ihrem Amiga. Mit dem Amiga-Musikbuch – ein Buch, das voller Musik steckt. Hier werden Sie zu einem Komponisten ausgebildet, der nicht nur die notwendigen Grundbegriffe der Musiktheorie beherrscht, sondern auch modernste Technik einzusetzen weiß. Denn in diesem Buch erfahren Sie alles zu den Musikprogrammen Sonix, DeLuxe Construction Set und Audio Master. Dabei lernen Sie auch, wie Sie Sound-Sampler und MIDI-Interface professionell einsetzen. Wenn Sie mit diesem Buch gearbeitet haben, sollten Sie auch gleich der GEMA beitreten.
Amiga-Musikbuch
Hardcover, 384 Seiten, DM 49,–

COUPON

Bitte einsenden an:
DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf

HIERMIT BESTELLE ICH FÜR MEINEN AMIGA

NAME, VORNAME

STRASSE

ORT

zzgl. DM 5,– Versandkosten unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

EDWORK

Das AMIGA-Werkzeug.

NEU !

Profi-Programmierer können EDwork nur bedingt empfehlen:
Erst dachten sie, EDwork sei nur irgendeiner der zahlreichen, mehr oder weniger nützlichen Editoren für den Amiga. Dann stellten sie fasziniert fest, was in dem neuen Programm steckt. Und schließlich wurde ihnen schlagartig klar, daß mit EDwork der talentierte Nachwuchs mit Macht nach oben drängen wird. Denn nach Lektüre des ausführlichen Handbuchs ist EDwork das Werkzeug für alle, die ihren ganz individuellen Editor programmieren wollen, um damit universell arbeiten zu können.

Hier sind einige der Punkte, die Profis lieber für sich behalten möchten: Mit über 120 Befehlen (und einem dazu mitgelieferten Compiler, der die Programme schneller macht) gestalten Sie sich Ihre ganz persönliche EDwork-Version – etwa mit eigenen Pulldown-Menüs. Über 65.000 Makros sind definierbar. Das Tasteninitialisierungs-Programm schreiben Sie nach Wunsch und legen damit beispielsweise kleine Programme auf einzelne Funktionstasten.

Effizient

Mitgelieferte EDwork-Hilfsprogramme für C- und Assembler-Programmierer machen Ihre Arbeit effizienter: Sie komprimieren, verschönern oder korrigieren Ihre Quelltexte. Mit "SHIFT/F10" können Sie z. B. Ihr Programm aus dem Editor heraus kompilieren lassen (etwa durch Aufruf des Aztec-C-Compilers): Eine Fehlerdatei wird geladen und der erste Fehler angezeigt. Natürlich läßt sich EDwork auch an alle anderen Programmiersprachen anpassen.

Übersichtlich



Einer der augenfälligsten Pluspunkte von EDwork wird aber die Übersichtlichkeit Ihrer Programme sein. Durch die sogenannte Faltentechnik lassen Sie Module, die zur Zeit nicht Zeile für Zeile angezeigt werden sollen, einfach in den „Falten" verschwinden. Diesen schicken „Outline"-Effekt können Sie selbstverständlich auch im Berufs- oder Privatleben nutzen. Legen Sie beispielsweise bei der Projekt- oder Terminplanung die Daten über benötigte Mitarbeiter oder Ressourcen einfach in den „Falten" ab: Aus den Augen, aber nicht aus dem Speicher.

Preiswert

Werden Sie jedoch hellhörig, wenn Kollegen, die sich schon eine EDwork-Version gekauft haben, ihre Neuanschaffung ebenso diskret vor Ihnen verbergen wollen wie weggefaltete Programmteile. Denn EDwork ist viel zu gut und viel zu preiswert, als daß seine Vorteile nur wenigen zugute kommen sollten.
EDwork gibt es für 99,– Mark. Der Editor läuft auf allen Amiga-Rechnern der 500er-, 1000er- und 2000er-Serie.

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

Bitte einsenden an: DATA BECKER
Merowingerstraße 30, 4000 Düsseldorf 1

☐ Hiermit bestelle ich EDwork Amiga für DM 99,–

☐ per Nachnahme ☐ Verrechnungsscheck anbei

NAME, VORNAME

STRASSE

ORT